НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Салій Д.В.
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Геометричне моделювання в інформаційних системах»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Розробка програмного забезпечення для автоматизації енергоменеджменту на підприємстві на платформі ОС Windows»

Виконала:
студентка IV курсу, групи ТР-61
Салій Дарина Валеріївна _____

Керівник:
професор, д.т.н.
Отрох Сергій Іванович _____

Рецензент:
старший викладач, к.т.н.
Зенів Ірина Онуфріївна _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України
„Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Олександр Коваль
(підпис)

”___”_____2020 р.

ЗАВДАННЯ
на дипломну роботу студенту

Салій Дарина Валеріївна
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка програмного забезпечення для автоматизації енергоменеджменту на підприємстві на платформі ОС Windows

керівник роботи професор, д.т.н. Отрох Сергій Іванович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1268-с**

2. Строк подання студентом роботи 10 червня 2020

3. Вихідні дані до роботи: персональний комп’ютер під керуванням операційної системи Microsoft Windows, мова програмування Java, середовище розробки IntelliJ IDEA, фреймворк Apache Maven.

4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) проаналізувати існуючі програмні рішення та можливі засоби реалізації взаємодії, обґрунтувати обрані програмні застосунки та шляхи розробки програмних додатків, розробити програмне забезпечення, розробити користувацький інтерфейс, зробити висновки за результатами роботи

5. Перелік ілюстраційного матеріалу

2. Аналіз існуючих систем 3. Засоби реалізації 4. Опис програмної реалізації системи 5. Методика роботи користувача з програмою

6. Дата видачі завдання ” 1 ” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	01.10.19	
2.	Вивчення та аналіз задачі	11.02.20	
3.	Розробка архітектури та загальної структури системи	22.04.20	
4.	Розробка структур окремих підсистем	29.04.20	
5.	Програмна реалізація системи	03.05.20	
6.	Оформлення пояснювальної записки	10.05.20	
7.	Захист програмного продукту	15.05.20	
8.	Передзахист	10.06.20	
9.	Захист	17.06.20	

Студент _____ Салій Д.В
(підпис) (прізвище та ініціали)

Керівник роботи _____ Отрох С.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Обсяг дипломної роботи складає 54 сторінки, 36 рисунків. 3 додатки і 10 посилань

Метою дипломної роботи є розробка програмного додатку, який дозволить користувачу вести облік електроенергії та запровадити політику енергоменеджменту на підприємстві.

Під час роботи над дипломним проектом було проаналізовано предметну область енергоменеджменту, стандарти систем енергоменеджменту та описано існуючу систему.

Для розробки додатку було використано мову програмування Java, базу даних створено засобами мови SQL та SQL Server. Інтерфейс налаштовано на платформі JavaFX за допомогою мов FXML та CSS.

Результатом роботи стало створення програмного продукту для моніторингу використання електроенергії на підприємстві.

Ключові слова: моніторинг, аналіз, використання електроенергії, енергоменеджмент, Java, SQL.

ABSTRACT

The amount of thesis is 54 pages, 36 pictures, 3 applications and 10 references

The purpose of the project is to develop a software application that will allow the user to keep records of electricity at the enterprise and to conduct energy management policy.

During the work on the diploma project the subject area of energy management, standards of energy management systems were analyzed and the existing system was described.

The Java programming language was used to develop application, and the database was created using the SQL language and SQL Server. The interface is configured on the JavaFX platform by using FXML and CSS languages.

The result was the creation a software product for monitoring using of electricity at the enterprise.

Keywords: monitoring, analysis, electricity use, energy management, Java, SQL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

JVM — Java Virtual Machine

SQL — Structured Query Language

CSS — Cascading Style Sheets

XML — eXtensible Markup Language

HTML — HyperText Markup Language

ISO — International Organization for Standardization

ВСТУП

На сьогоднішній день підприємствам необхідна система енергоменеджменту, адже вона дозволяє оптимізувати витрати електроенергії та фінансові затрати підприємства.

Енергоменеджмент — це спосіб управління енергоспоживанням на підприємстві, що дозволяє значно оптимізувати обсяги енерговитрат. Основним інструментом енергоменеджменту є енергетичний аудит. Світова практика показує, що підвищення енергоефективності досягається здебільшого за рахунок організаційних змін в системі управління енергогосподарством, тобто за рахунок поліпшення системи енергоменеджменту. Тому, для вирішення питань підвищення енергоефективності на підприємствах вводиться система енергоменеджменту. Система енергоменеджменту — це комплекс взаємопов'язаних і взаємодіючих елементів підприємства, спрямований на формування енергетичної політики підприємства, постановку цілей, розробку заходів по досягненню цих цілей.

Енергоменеджмент може забезпечити:

- виявлення дефектів, поганої роботи і збоїв в енергоспоживаючих системах;
- швидке втручання у разі несприятливих тенденцій до збільшення використання енергоресурсів;
- визначення рекомендованих удосконалень і їх пріоритет;
- більш уважне ставлення до питань використання енергії.

Одними з найважливіших завдань дипломної роботи є розробка і створення програмного забезпечення для обліку використання електроенергії на підприємстві. Для вирішення даної проблеми було поставлено завдання розробити додаток, який надав би змогу користувачу відслідковувати витрати електроенергії на підприємстві, і надавав би детальну інформацію по кожному споживачу електроенергії.

Під час виконання дипломного проекту було створено програмний продукт, що має базу даних та в якому користувач має змогу переглядати детальні звіти по

енерговитратам на підприємстві, у конкретному відділі та по витратам електроенергії споживачами на підприємстві.

Для програмної реалізації було вирішено використати мову програмування Java. Для автоматизації збірки проекту було використано технологію Maven. Для побудови архітектури бази даних було використано систему управління базами даних MySQL та засоби мови SQL. Для розробки інтерфейсу було використано мову FXML та CSS на платформі JavaFX.

Особливу увагу приділено можливості розширення функціоналу даної системи сторонніми розробникам, для цього були використані технології, що стандартизують підхід до написання коду.

1. ЗАДАЧА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ АВТОМАТИЗАЦІЇ ЕНЕРГОМЕНЕДЖМЕНТУ НА ПІДПРИЄМСТВІ

Перед виконанням дипломної роботи була поставлена задача організувати систему автоматизації обліку витрат електроенергії, контролю використання електроенергії кожним споживачем на будь – якій мові програмування.

Призначення програмного продукту полягає в:

— можливості додавати нових споживачів

— можливості додавати нові відділи підприємства

— забезпеченні внесення змін в існуючу базу даних

— зберіганні інформації про споживачів та відділи підприємства

— формування звіту про використання електроенергії на підприємстві

— формуванні звіту про використання електроенергії у відділі

Сьогодні існує декілька прикладних програм для обліку електроенергії, але жодна з них не задовольняє повністю необхідним умовам. В дипломній роботі буде продемонстровано використання власного програмного продукту для автоматизації енергоменеджменту. Також для поліпшення роботи системи було поставлено задачу спроектувати та розробити базу даних.

Програмний продукт повинен мати такі функції:

— додавання та перегляд відділів підприємства

— додавання та перегляд споживачів електроенергії

— редагування та видалення відділів підприємства

— редагування та видалення споживачів електроенергії

— перегляд звіту про використання електроенергії на підприємстві

— перегляд звіту про використання електроенергії у відділі

2. АНАЛІЗ СИСТЕМ АВТОМАТИЗАЦІЇ ЕНЕРГОМЕНЕДЖМЕНТУ НА ПІДПРИЄМСТВІ

Україна входить в список країн із наявним дефіцитом енергоресурсів, а власні видобутки країни задовільняють лише близько п’ятидесяти відсотків від загальної потреби в енергоносіях. Зважаючи на це, доволі важливим напрямком політики держави у сфері розвитку економіки є привнесення у повсякденне життя країни поняття енергоменеджменту, яке включає в себе як принципи енергозбереження, так і принципи ефективного розподілу та споживання енергії. Значну частину розвитку даного напрямку економіки можуть зайняти автоматизовані системи енергоменеджменту, енергоаудиту та контролю за споживанням енергії, звільняючи споживачів енергії від необхідності вести облік енерговитрат вручну.

“Енергетичний менеджмент” — це діяльність по забезпеченню раціонального використання енергетичних ресурсів на підприємстві, що дозволяє значно оптимізувати обсяги енерговитрат.

2.1 Огляд існуючих систем обліку енергоресурсів

Існуючі програми для обліку енергоресурсів є досить дорогими і ресурсозатратними. Однією з таких систем є система eXpertPower від компанії SATEC. Система eXpertPower забезпечує технічний та комерційний облік, контроль поточного навантаження, контроль споживання, а також підтримує рішення при плануванні енергоспоживання та вироблення енергозберігаючої політики. Для забезпечення цих завдань компанія SATEC розробила прилад EM720, який зображено на рисунку 2.1.

Рисунок 2.1 — Прилад EM720

Прилад EM720 — багатофункціональний електронний прилад, який виконує функції високоточного та багатотарифного лічильника електроенергії, комерційного обліку, реєстрації аварійних подій [1]. Прилад EM720 дозволяє зменшити число елементів в системі, що дозволяє зменшити вартість, сприяє підвищенню надійності та простоти обслуговування системи.

Для поєднання приладів в систему обліку електроенергії компанія SATEC розробила програмне забезпечення eXpertPower — веб-додаток, який в режимі реального часу дає користувачам змогу отримати всі дані про стан електричної мережі, використовуючи локальний або віддалений доступ [1]. На рисунку 2.2 зображено діаграму системи eXpertPower.

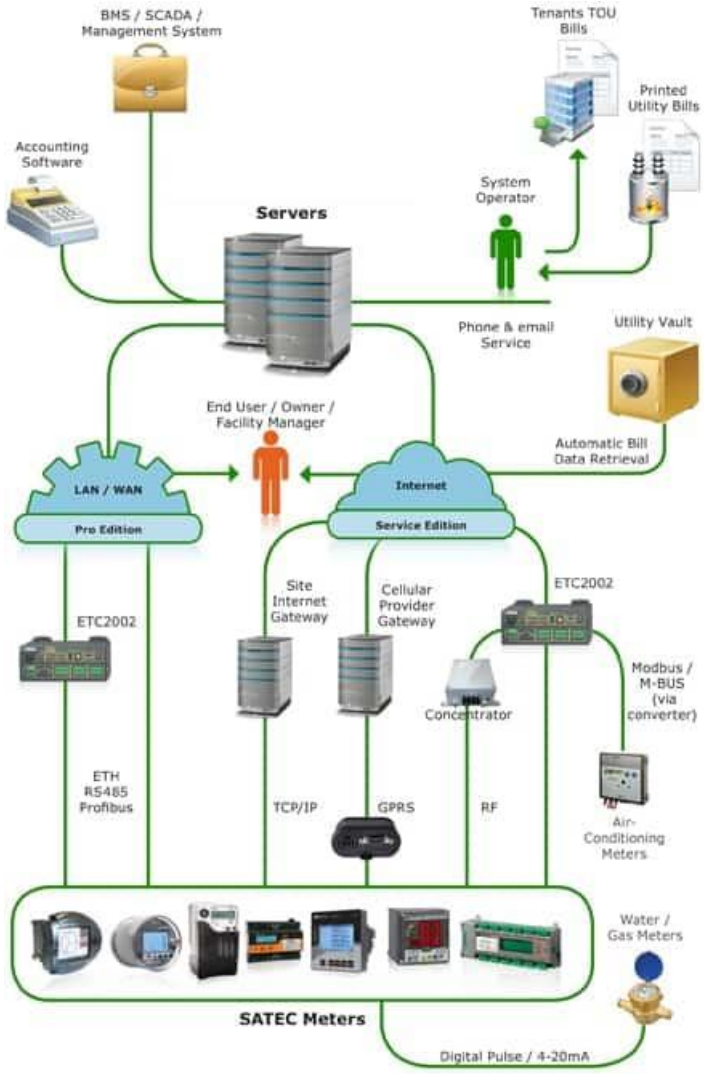

Рисунок 2.2 — Діаграма системи eXpertPower

На рисунку 2.3 продемонстровано інтерфейс додатку eXpertPower, який використовує стандартний веб-браузер в якості інтерфейсу для користувача. Таке рішення дозволяє зменшити вартість експлуатації системи, оскільки відсутня необхідність в спеціальному програмному забезпечені та обладнанні.

Рисунок 2.3 — Інтерфейс системи eXpertPower

Недоліками системи eXpertPower є велика кількість задіяних у процес людських ресурсів, апаратури, а також велика вартість компонентів, налаштування і обслуговування.

2.2 Стандарти енергоменеджменту

Енергоменеджмент — це постійно діюча система керування енергоспоживанням, що дає можливість значно оптимізувати енерговитрати та запобігти надмірним втратам енергії на виробництві чи, в загальному випадку, на об’єкті господарювання. Крім того, ефективна система енергоменеджменту дозволяє прогнозувати та контролювати процес вироблення, транспортування та споживання енергії для об’єкту господарювання.

Система енергоменеджмента являє собою комплекс організаційних заходів, технічних засобів та програмного забезпечення, підсистеми якого працюють у

взаємозв’язку один з одним, з метою формування підходу до енергетичної політики підприємства чи виробництва, вироблення оптимальної стратегії зниження енерговтрат, та максимальної оптимізації енерговитрат [2].

2.2.1. Стандарт ISO 50001

ISO (англ. International Organization for Standartization) – міжнародна організація, що займається створенням та випуском стандартів.

Стандарт ISO 50001 – міжнародний стандарт, створений Міжнародною організацією стандартизації (ISO) для систем керування енергосистемами та енергоресурсами, що визначає вимоги для встановлення, впровадження та покращення систем енергоменеджменту, що дозволяє організаціям слідувати систематичному підходу до для досягнення найбільш оптимального розподілення та використання енергоресурсів, тобто, власне, досягнення цілі ефективного енергоменеджменту [3].

Стандарт ISO 50001 надає організації повноцінну стратегію заходів в області менеджменту та впровадження технічних засобів для підвищення ефективності енергосистеми об’єкту господарювання. Більш того, даний стандарт легка маштабується, та є універсальним для будь-якого об’єкту господарювання або організації, не залежачи від форми власності (державна чи приватна), місця знаходження чи розміру, тобто стандарт ISO 50001 є універсальним алгоритмом досягнення енергоефективності організації.

2.2.2 Стандарт EN 16001:2009

Стандарт енергоуправління EN 16001:2009 був розроблений в Великобританії Європейським комітетом по стандартизації із залучення спеціалістів із Британського інституту стандартів (англ. British Standards Institution, BSI). Даний стандарт є прямим предком описанного вище міжнародного стандарту ISO 50001. Більшість

підходів ISO 50001, таких як надання повної та універсальної стратегії оптимізації енергоспоживання, а також більшість його рекомендацій були закладені ще у стандарті EN 16001:2009, де “2009”-це рік прийняття стандарту.

EN 16001 так само як і ISO 50001, встановлює принципи керування енергосистемою промислових підприємств, комерційних організацій та об’єктів господарювання, включно із об’єктами державної форми власності.

Обидва вищеописані стандарти сприяють підвищенню енергоефективності підприємств, виробництв та організацій, а також надають менеджерському складу організацій інструменти для побудови енергетичної політики та модернізації виробництва.

У стандарт EN 16001 так само як і в ISO 50001 було закладено принцип масштабованості та універсальності, завдяки чому даний принцип, не зважаючи на наявність більш сучасного наступника, продовжує застосовуватися на об’єктах господарювання.

2.3 Висновки до розділу

У даному розділі було описано існуючі системи обліку електроенергії, а саме система eXpertPower. Було наведено особливості роботи, переваги і недоліки цієї системи. Також були розглянуті існуючі стандарти енергоуправління, а саме стандарт ISO 50001 і EN 16001:2009, які визначають вимоги для встановлення, впровадження та покращення систем енергоменеджменту.

3. ЗАСОБИ РОЗРОБКИ

3.1. Мова програмування Java

Для реалізації автоматизації енергоменеджменту було вирішено використовувати кросплатформну мову програмування Java, оскільки вона об’єктно-орієнтована, має зрозумілу документацію, велику кількість бібліотек і фреймворків та гарну швидкодію при високих навантаженнях.

Java є мовою розробки майже для всіх типів додатків і платформ. Java загальний стандарт для розробки додатків на мобільній платформі, ігор, веб-додатків та корпоративного програмного забезпечення [4]. Програми на Java транслюються у байт-код Java , який виконується віртуальною машиною Java, скорочено JVM. Окрім незалежності мови від операційної системи та обладнання у мови Java є ще одна важлива особливість, а саме гнучка система безпеки, у рамках якої виконання програмного коду контролюється повністю віртуальною машиною JVM, тобто будь-які операції, що перевищують встановлені повноваження програми відразу визивають переривання.

Основні можливості мови програмування Java:

— Автоматичне керування пам’яттю

— Більш розширені можливості обробки виключень

— Вбудовані в мову засоби створення багатопотокових додатків

— Уніфікований доступ до бази даних

— Вбудовані можливості функціонального програмування

— Висока продуктивність виконання

— Незалежність від архітектури

Для написання програмного коду на мові Java було обрано середовище програмування IntelliJ IDEA. IntelliJ IDEA - це інтегроване середовище розробки (IDE) для мов JVM, розроблене для досягнення максимальної продуктивності розробника [9].

3.2. Декларативна мова програмування SQL

База даних була побудована засобами мови SQL — декларативна мова програмування, яка застосовується для створення, модифікації та управління даними в реляційній базі даних, за допомогою системи управління базами даних MySQL.

Мова SQL являє собою, насамперед, інформаційно-логічну мову, яка призначена для опису, зміни та вилучення даних із реляційної бази даних [5]. Мова SQL представляє собою сукупність інструкцій, операторів та обчислюваних функцій.

Оператори мови SQL поділяються на:

1. Оператори визначення даних або DLL-оператори (CREATE, ALTER, DROP);

2. Оператори маніпуляції даними або DML-оператори (SELECT, INSERT, UPDATE, DELETE);

3. Оператори визначення режиму доступа до даних або DCL-оператори (GRANT, REVOKE, DENY);

4. Оператори контролювання транзакцій або TCL-оператори (COMMIT, ROLLBACK, SAVEPOINT);

Для підключення до бази даних було використано інтерфейс JDBC (англ. Java DataBase Connectivity) — платформенно незалежний промисловий стандарт взаємодії Java-додатків з різними СУБД [6].

Слід зазначити, що мова SQL реалізує декларативну парадигму програмування, тобто кожний оператор описує тільки необхідну дію, а СУБД вирішує як її виконати, тобто планує елементарні операції, які необхідні для виконання дії і виконує їх.

3.3. Фреймворк Apache Maven

Побудовано проект за допомогою фреймворка Apache Maven, який надає широкий спектр інструментів для підключення сторонніх бібліотек та плагінів до проекту лише через опис даних підключень у конфігураційному файлі “pom.xml”, що дозволяє перейти від імперативного методу конфігурації збірки, до більш зручного, декларативного [7].

У файлах опису проекту міститься його специфікація, а не окремі команди виконання. Всі задачі по обробці файлів Maven виконує за допомогою обробки послідовності вбудованих і зовнішніх плагінів.

Життєвий цикл Maven-проекту це список фаз, які визначають порядок дій при побудові проекту. У життєвого циклу Maven є три незалежні порядки виконання:

— Clean – цикл для очищення проекту.

— Default – основний життєвий цикл, який поділяється на наступні фази:

- Validate — перевірка структури проекту
- Generate-sources;
- Process-sources;
- Compile — компіляція вихідних текстів;
- Test — тестування коду;
- Package — пакування скомпільованих классів;
- Install — установка програми в локальний репозиторій;
- Deploy — установка програми на віддалений репозиторій;

— Site – цикл генерації проектної документації.

Стандартні життєві цикли можуть бути доповнені функціоналом з допомогою Maven-плагінів. Плагіни дозволяють додавати до стандартного циклу нові кроки або розширювати існуючі кроки.

3.4. Програмна платформа JavaFX

Інтерфейс вікон програми розроблено на платформі JavaFX — це програмна платформа для створення настільних додатків, які можуть працювати на широкому спектрі пристроїв [8].

JavaFX надає більш великі можливості порівняно з іншими подібними платформами, а саме великий набір елементів управління, можливість роботи с мультимедіа, графікою, а також декларативний спосіб опису інтерфейсу за допомогою мови розмітки FXML, можливість налаштування стилю інтерфейсу за допомогою CSS.

Платформа JavaFX міститься в декількох модулях, які зображено на рисунку 3.1.

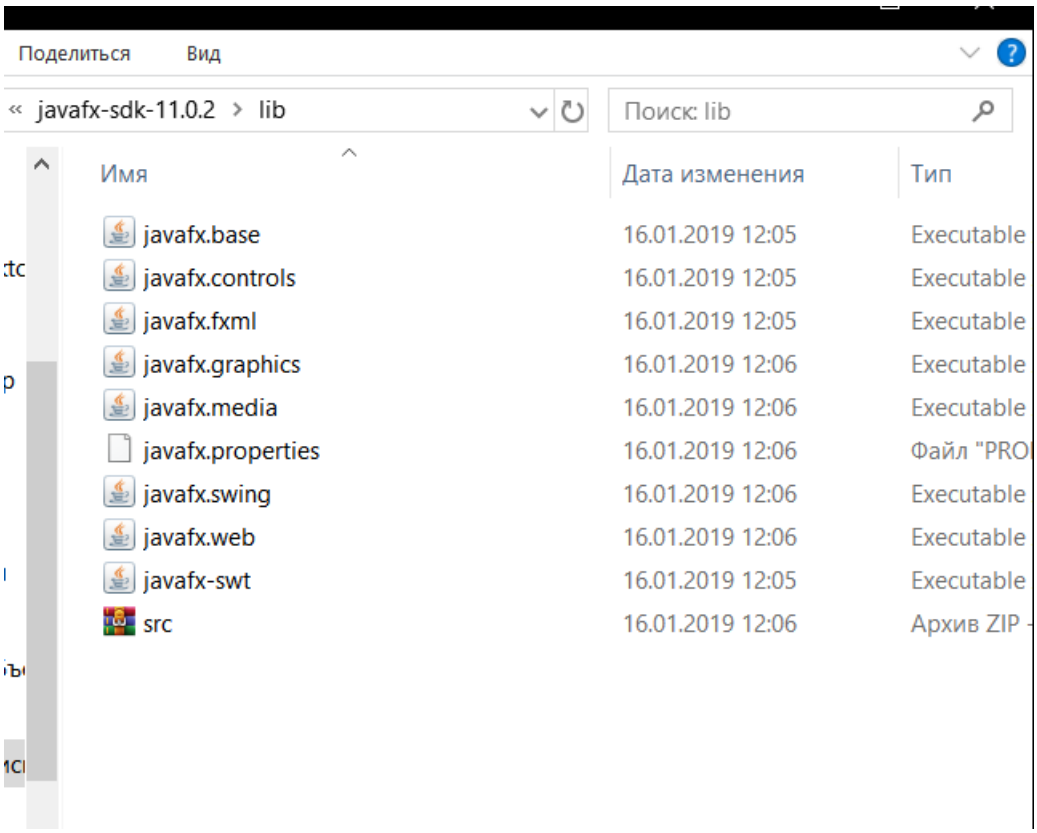

Рисунок 3.1 — Модулі платформи JavaFX

— javafx.base — визначає базовий функціонал фреймворка
— javafx.controls — визначає елементи управління, діаграми;
— javafx.fxml — визначає функціонал для роботи з FXML;
— javafx.graphics — визначає функціональність вікон, можливості малювання, анімації, користувацького вводу;
— javafx.media — визначає функціонал для роботи з мультимедія;
— javafx.swing — визначає інтерфейс для взаємодії з фреймворком Swing
— javafx.web — визначає функціонал WebView.

3.5. Мова розмітки CSS

Налаштовано інтерфейс за допомогою мови CSS — формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки [10].

Мова розмітки CSS використовується для візуальної презентації сторінок, які написані мовою HTML та XHTML, а також для інших видів XML-документів. CSS використовують для задання стилів, кольорів, шрифтів, положення об’єктів на сторінці та інших аспектів зовнішнього вигляду. Особливістю мови CSS є можливість розділити зовнішній вигляд сторінки інтерфейсу та контент сторінки, написаний на XML мові. Така особливість мови позитивно впливає покращенню доступності та сприйняття контенту сторінки, забезпечує контроль та гнучкість відображення контенту в різних умовах, робить контент більш простим та структурованим, прибирає повтори, тощо.

XML документ може по-різному відображатись в залежності від використаного CSS стилю.

Основні переваги мови CSS:

— Інформація про стиль однієї або декількох сторінок міститься в одному .css-файлі, а тому дозволяє робити швидкі зміни в дизайні сторінок
— Можливість налаштувати різні стилі сторінки в залежності від умов, наприклад стиль для мобільних пристроїв, для друку, тощо.

— Розділення контенту сторінки від стилю сприяє більш структурованому написанню коду та простоті його сприйняття.
— Швидке завантаження сторінок і зменшення обсягу інформації, зменшене навантаження на сервер та канал передачі.
— Більш точний контроль над зовнішнім стилем сторінок

Основними недоліками CSS є:

— Різне відображення сторінки оформленої за допомогою CSS у різних браузерах, особливо застарілих, які по-різному інтерпретують дані CSS.
— Складна і проблемна техніка дизайну
— Часто зустрічається необхідність окрім .css-файлу виправляти і пов’язані з ним XML теги

3.6 Висновки до розділу

У даному розділі були розглянуті інструменти та технології розробки програмного продукту. Додаток було розроблено за допомоги об’єктно-орієнтованої мови Java у середовиці IntelliJ IDEA. У розробці інтерфейсу додатку було використано платформу JavaFX, мову FXML і мову розмітки CSS. Для збірка і підключення бібліотек до проекту було використано фреймворк Apache Maven. База даних для проекту створена засобами мови SQL у середовищі MySQL Workbench 8.0 CE.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ

4.1. Структура програмного забезпечення

Для реалізації задачі автоматизації енергоменеджементу на підприємстві був розроблений віконний додадок на операційній системі Windows . На рисунку 4.1 представлений інтерфейс головного вікна додадку.

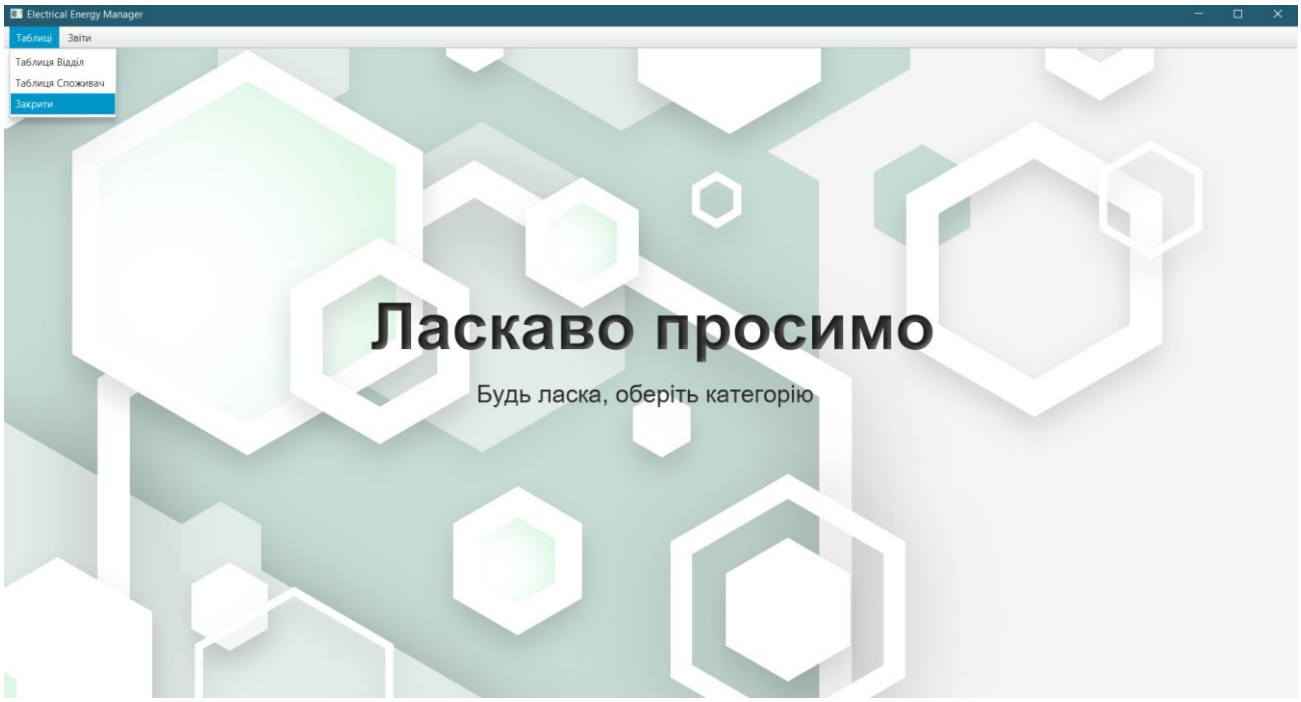

Рисунок 4.1 — Головне вікно програми

Для реалізації внесення даних про користувачів електроенергії у базу даних було розроблено окреме вікно “Таблиця Споживач” , що представлено на рисунку 4.2.

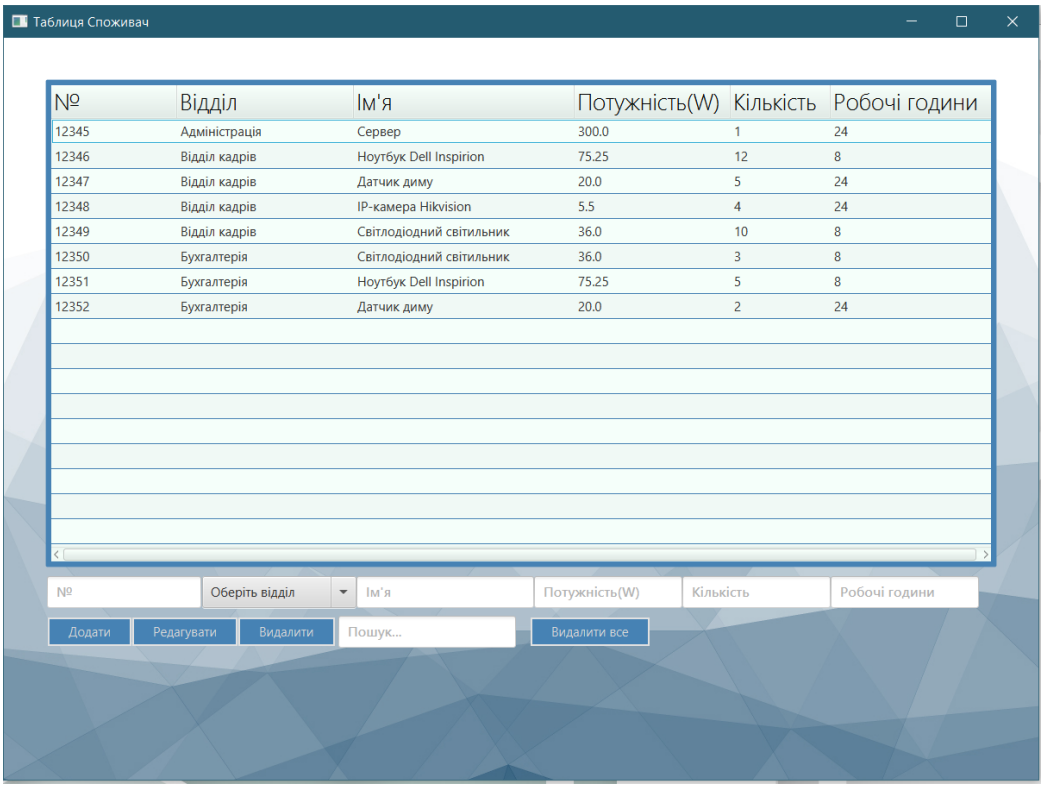

Рисунок 4.2 — Вікно “Таблиця Споживач”

Для здійснювання внесення даних про відділи підприємства у базу даних було розроблено окреме вікно “Таблиця Відділ”, що представлено на рисунку 4.3.

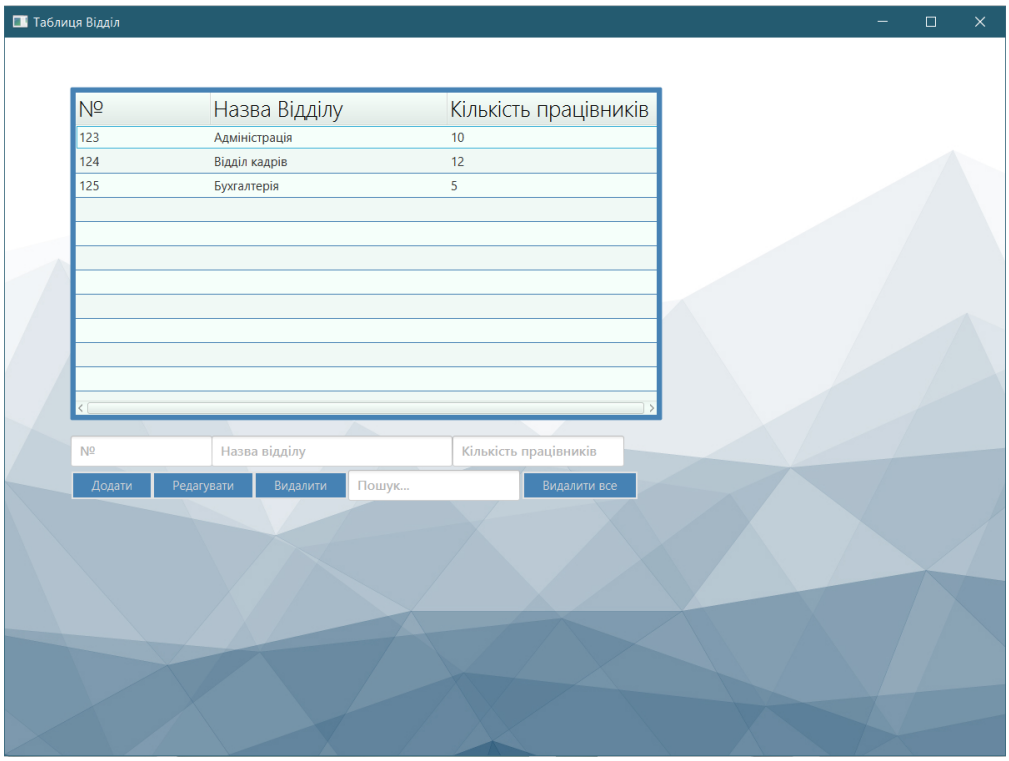

Рисунок 4.3 — Вікно “Таблиця Відділ”

При заповненні даними таблиці Відділ і таблиці Споживач з’являється можливість отримати звіт про витрати електроенергії у конкретному відділі та спостерігати на діаграмі скільки електроенергії витратив кожний споживач у відділі. На рисунку 4.4 зображено вікно звіту про енерговитрати у відділі.

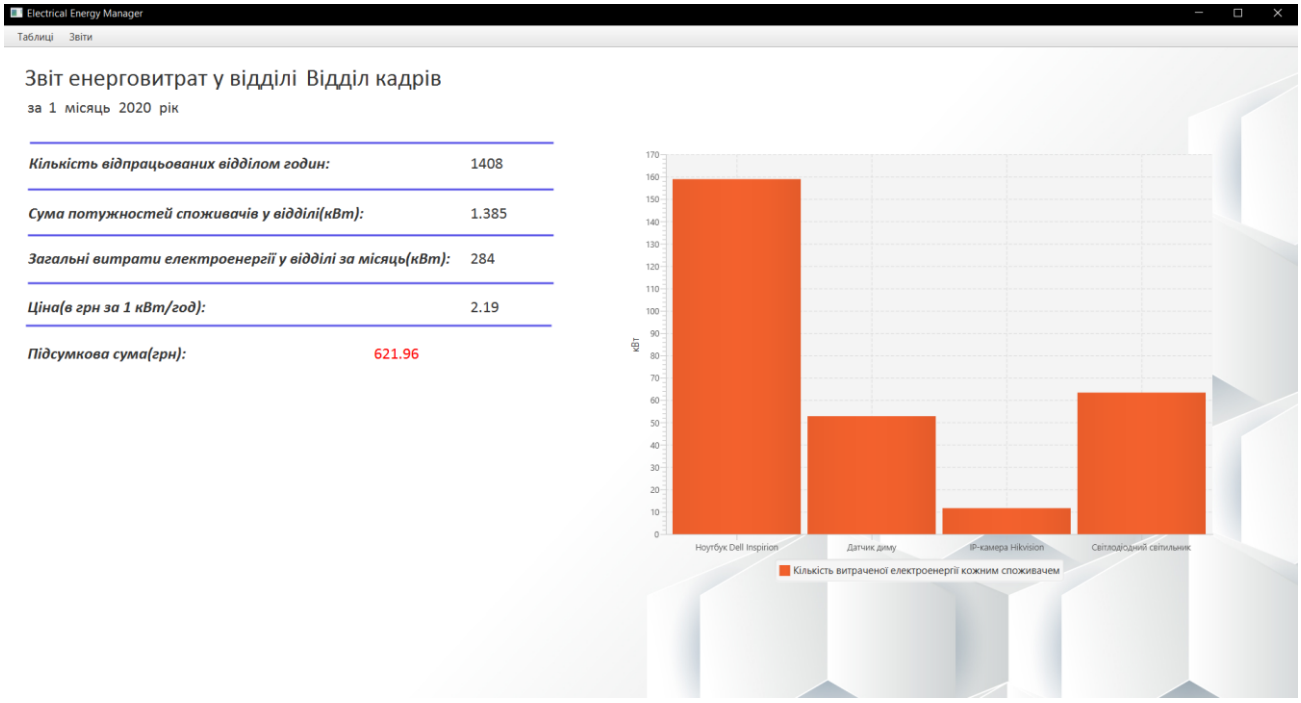

Рисунок 4.4 — Вікно “Звіт енерговитрат у відділі”

Також можна отримати загальний звіт, в якому міститься інформація про витрати електроенергії у всіх відділах та кругова діаграма розподілення енерговитрат у відділах. На рисунку 4.5 зображено вікно “Загальний звіт”.

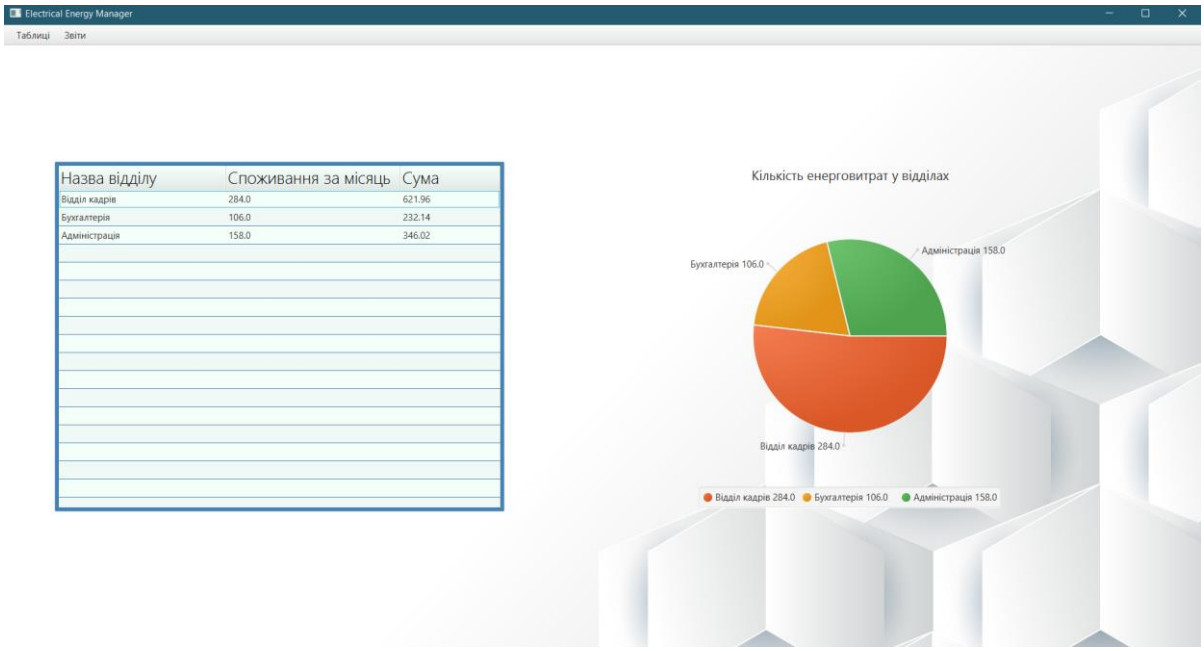

Рисунок 4.5 — Вікно “Загальний звіт”

Згідно вимог фреймворку Maven архітектура проекту має структуру папок, як показано на рисунку 4.6.

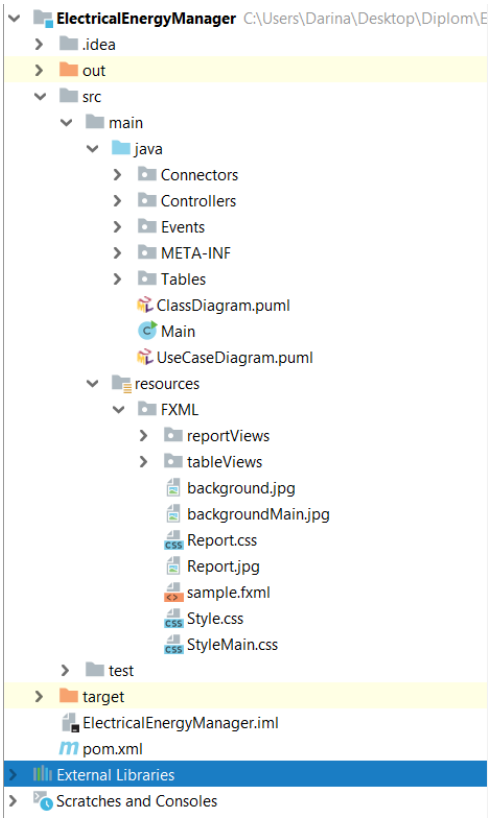

Рисунок 4.6 — Структура файлів проекту

Структура проекту складається з наступних модулей: src, target, .idea, External Libraries та файлів ElectricalEnergyManager.iml та pom.xml.

Модуль src поділяється на папки main та test, а main у свою чергу на java та resources. Папка src містить усю логіку програми, target – спеціальні файли, які необхідні для компіляції проекту. Каталог .idea — це створений автоматично самим середовищем InteliJ Idea каталог, який містить усю необхідну інформацію про проект, в тому числі файли метаданих [6], каталог External Libraries містить всі сторонні бібліотеки, що були підключені через конфігураційний файл “pom.xml”, тобто були додані до проекту за допомогою модуля-збірника Apache Maven. Файл pom.xml — це основний конфігураційний файл проекту, що декларує порядок збірки проекту для модулю Maven, у ньому знаходиться команди декларування на мові XML, завдяки файлу pom.xml підключаються сторонні бібліотеки, плагіни та визначається порядок компіляції модулів, а також задаються інші параметри збору проекту в самодостатню збірку.

Як вже зазначалося раніше, модуль src містить main, який поділяється ще на два модулі:

— папка java містить код організації роботи з базою даних та інтефейсом користувача, а також алгоритм логіки програми.

— папка resources містить файли розширення .fxml та .css для налаштування інтерфейсу користувача.

4.2 Діаграма класів проекту

Діаграма класів — це структурна діаграма мови моделювання UML, яка зображує класи, типи даних, методи системи, їх зміст та відношення. Діаграма класів застосовується для візуалізації, документування та для конструювання шляхом прямого та зворотного проектування.

Ціллю створення діаграми класів є графічне представлення статичної структури декларативних елементів системи, таких як класи, типи, методи, тощо.

На рисунку 4.7 відображено діаграму класів створеного програмного продукту

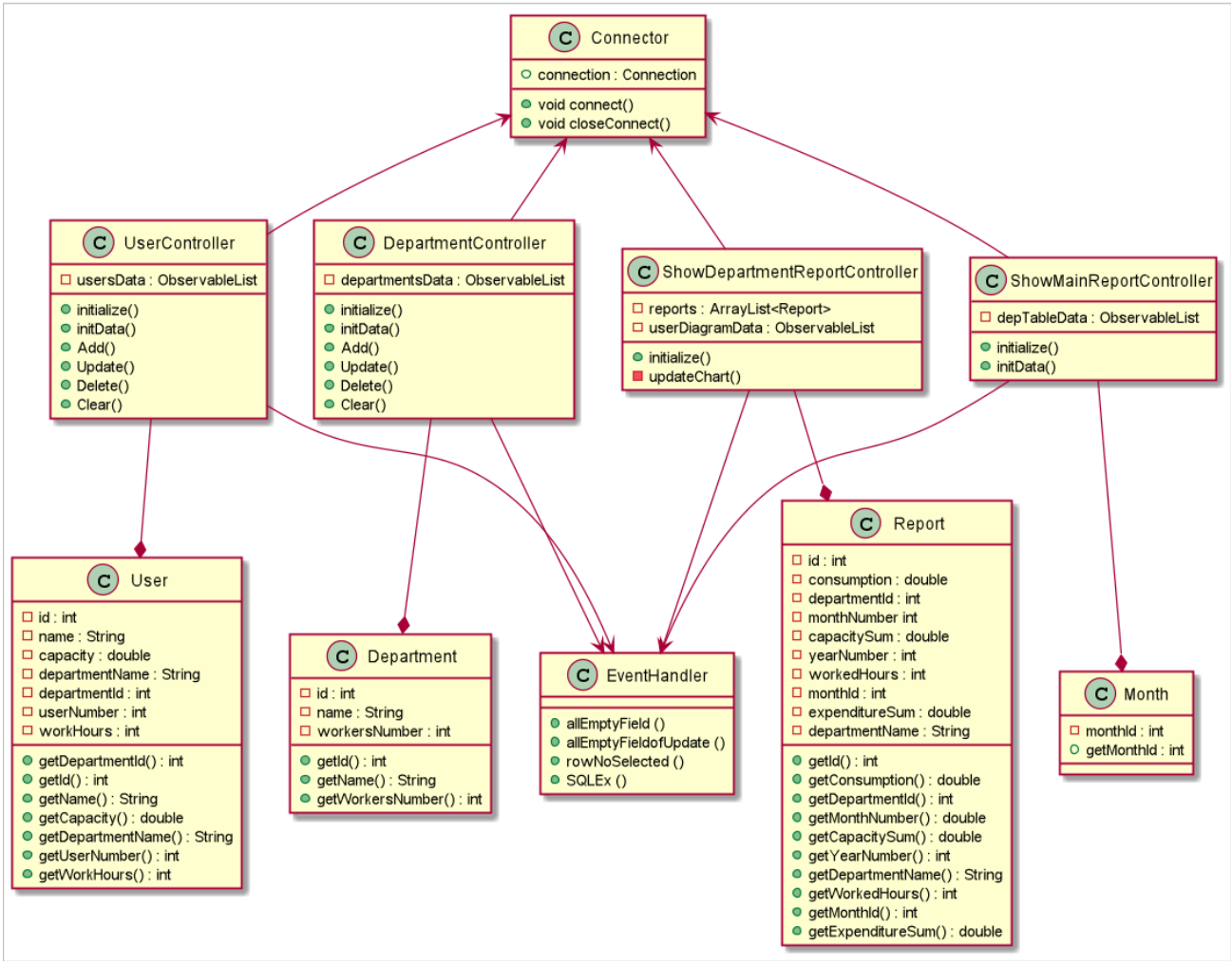

Рисунок 4.7 Діаграма класів додатку

На діаграмі класів класи представлені у вигляді рамок, в яких міститься три компонента. У верхній частині назва класу, всередині поля класу, у нижній частині методи класу.

Для зображення логічних відношень між об’єктами класів на діаграмі використовується декілька видів взаємозв’язків.

Асоціація показує, що об’єкти одного класу зв’язані з об’єктами іншого класу таким чином, що можна переміщатися від об’єктів одного класу до об’єктів іншого класу. На діаграмі класів асоціація в одну сторону зображується суцільною лінією з стрілкою на кінці, двостороння асоціація зображується суцільною лінією.

Агрегація — це різновид асоціації, яка являє собою структурний зв’язок між цілим і його частинами. Агрегація застосовується коли один клас є колекцією чи

контейнером інших. На діаграмі агрегація зображується лінією з порожнім ромбом на блоці класу.

Композиція — більш суворий варіант агрегації. Композиція має залежність від часу існування екземплярів класу-контейнеру і екземплярів класів, що містяться в ньому. Графічно композиція зображується як агрегація, але з заповненим ромбом.

Для зображення взаємозв’язків між класами системи використовується такі типи зв’язку як: успадкування, імплементація та залежність.

Успадкування демонструє що один із двох пов’язаних класів являється частковою формою іншого класу. Графічно успадкування зображується лінією з порожнім трикутником.

Імплементація або реалізація — це відношення між двома елементами моделі, в якому один елемент реалізує поведінку, яка задана іншим. На діаграмі зображується як успадкування, але пунктирною лінію

Залежність — це форма відносин використання, в якій зміна в специфікації одного спричиняє зміну іншого. На діаграмі класів зображується штриховою лінією зі стрілкою, направленою від залежного елемента.

4.3 Use Case діаграма проекту

Use Case діаграма — це опис поведінки системи, яка взаємодіє з кимось або чимось із зовнішнього середовища. Діаграми Use Case використовуються для аналізу користувацьких та функціональних вимог до системи.

На рисунку 4.8 зображено Use Case діаграма створеного програмного забезпечення.

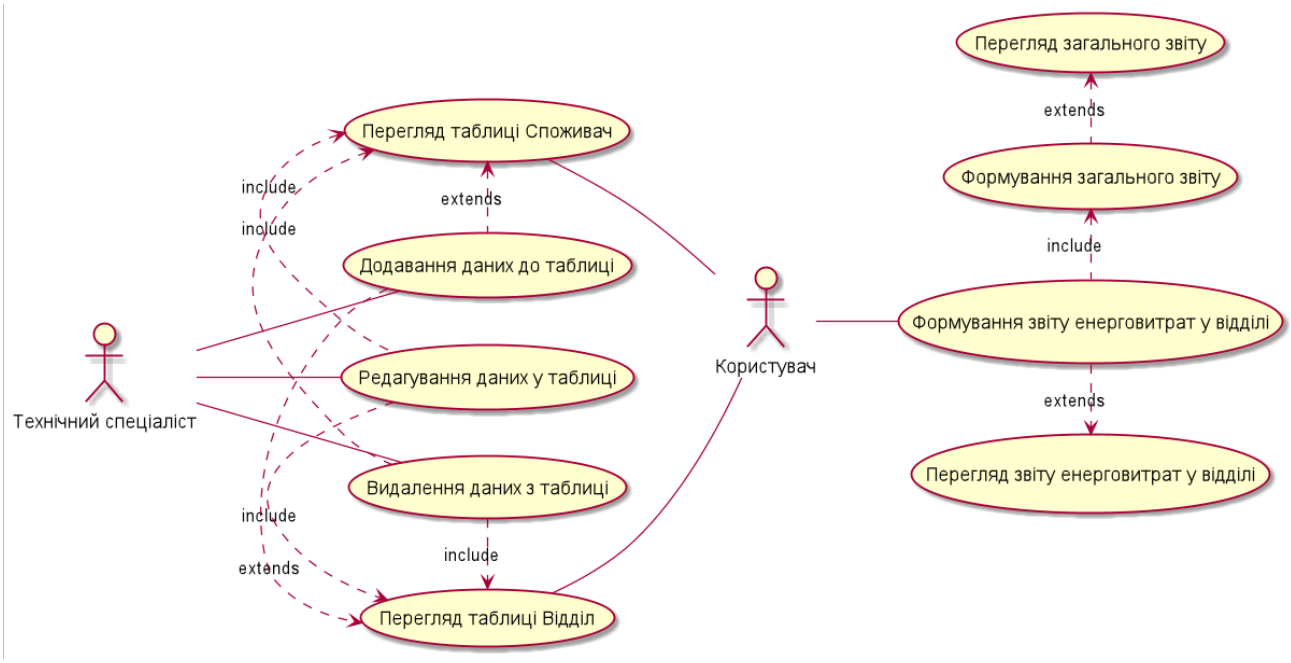

Рисунок 4.8 Use Case діаграма додатку

4.4 Опис структури бази даних

Під час розробки програми виникла потреба у створенні бази даних, для того щоб зберігати інформацію про відділи підприємства, про споживачів електроенергії у цих відділах, а також для оперативного редагування інформації і зберігання звітів в системі протягом року для аналізу споживання електроенергії підприємством за рік, і для оптимізації споживання електроенергії на наступній рік. На рисунку 4.9 зображено схему створеної бази даних.

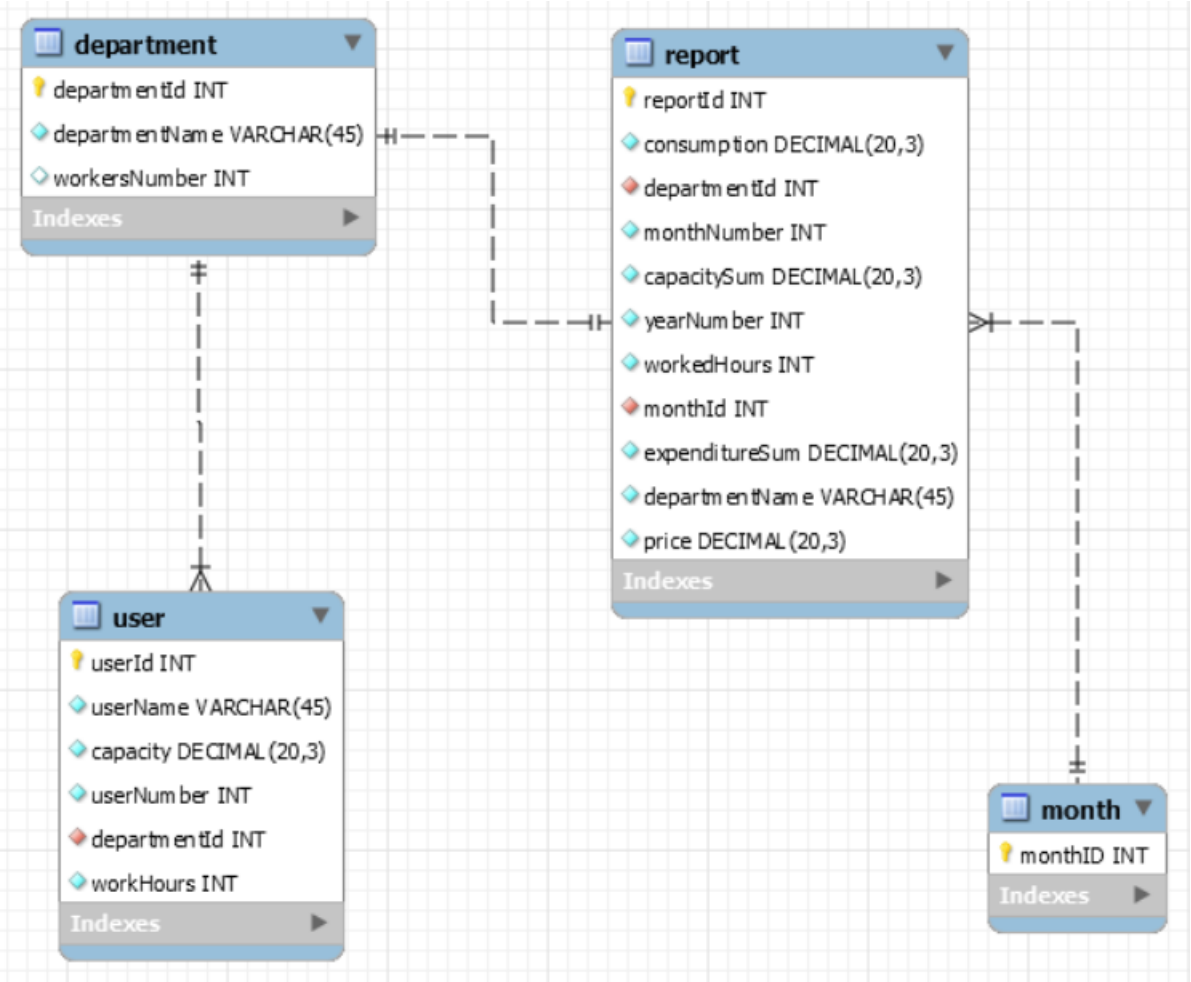

Рисунок 4.9 — Схема бази даних

У створеній базі даних зберігається інформація про кожний відділ, а саме його номер, назву та кількість працівників.

У таблиці Споживач міститься інформація про номер споживача, його назву, потужність, номер відділу в якому знаходиться споживач, кількість екземплярів споживача у відділі та кількість годин роботи користувача.

У таблиці Звіт зберігаються дані про номер звіту, його назва, номер місяця, сума спожитої за місяць електроенергії, номер відділу, сума потужностей споживачів електроенергії у відділі, номер місяця, рік, ціну електроенергії за місяць, підсумкову суму, а також кількість відпрацьованих відділом годин.

У таблиці “Місяць” міститься поле “monthId”, яке є первинним ключем. Таблиця “Місяць” створена для зберігання звітів для різних відділів за місяць.

Таблиця Відділ пов’язана із таблицею Звіт зв’язком типу “один до одного”,

тобто один запис із таблиці Відділ пов’язаний лише із одним записом таблиці Звіт.

Таблиця Відділ з таблицею Споживач пов’язана зв’язком “один до багатьох”, тобто один запис із таблиці Відділ пов’язаний з декількома записами із таблиці Споживач.

Аналогічно пов’язані таблиці Звіт та Місяць.

4.5 Опис алгоритму внесення даних у базу даних

Для зберігання інформації про підприємство та енерговитрати підприємства була розроблена база даних, що працює на сервері SQL Server 2017. Для взаємодії з створеною базою даних реалізовано клас Connector, в якому завдяки класу Connection створено connectionString в якій відбувається підключення до бази даних. На рисунку 4.10 продемонстровано строку підключення до бази даних у програмі.

```
public  void connect(){
    String connectionString = "jdbc:mysql://localhost/mydb?serverTimezone=Europe/Moscow&useSSL=false";
    String user = "root";
    String password = "1577";
    try {
        connection = DriverManager.getConnection(connectionString, user, password);
    } catch (SQLException e) {
        e.printStackTrace();
    }
```

Рисунок 4.10 Строка підключення до бази даних

Для внесення інформації в базу даних розроблено спеціальні вікна “Таблиця Відділ” і “Таблиця Споживач”. Для нормальної роботи з базою даних користувачу потрібно спочатку ввести дані у таблицю у вікні “Таблиця Відділ”. Користувачу потрібну ввести ідентифікатор відділу, його назву, та кількість працівників, які працюють в цьому відділі. Дані введені користувачем зчитуються програмою та вносяться у базу даних, таблиця у вікні “Таблиця Відділ” автоматично оновлюється при внесенні змін. У вікні “Таблиця Відділ” реалізовано кнопки “Додати”, “Редагувати”, “Видалити”, “Видалити все”, а також реалізовано пошук по таблиці.

Кнопки взаємодії з таблицею і навігацію по таблиці зображено на рисунку 4.11.

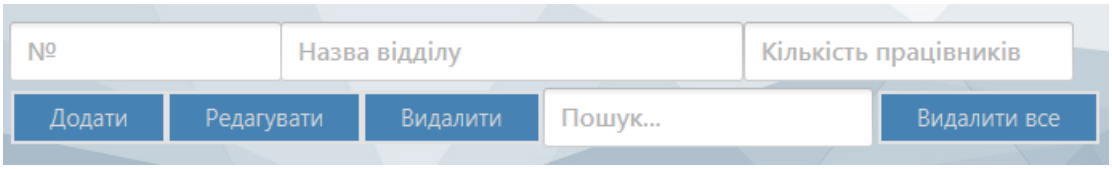

Рисунок 4.11 Кнопки взаємодії з таблицею і навігація по таблиці

Кнопка “Додати” добавляє до таблиці та у базу даних введений запис, для коректного додавання створено вікно попередження про те, що всі поля повинні буди заповнені, а також вікно попередження про те що користувач намагається додати вже існуючий ідентифікатор. Після коректного додавання програма виконує запит “insert” у базу даних до реалізованої там таблиці department.

Кнопка “Редагувати” дозволяє користувачу редагувати вже внесений запис у таблиці та у базі даних. Для редагування користувачу потрібну вибрати потрібний запис у таблиці, ввести необхідні зміни у полях вводу та натиснути кнопку “Редагувати”. Після цього програма виконує запит “update” у базу даних до реалізованої там таблиці department.

Кнопка “Видалити” реалізована для того щоб надати користувачу можливість видалення вже існуючих записів із таблиці та із бази даних. Для видалення користувачу необхідно вибрати запис у таблиці та натиснути “Видалити”. Для коректного видалення створено вікно попередження про те, що користувач намагається видалити зв’язаний з іншою таблицею запис. Після коректного видалення програма виконує запит “delete” у базу даних до реалізованої там таблиці department.

Кнопка “Видалити все” створена для того щоб при натисненні швидко видалити одразу всі записи у таблиці та у таблиці department у базі даних. Для коректної очистки таблиці створено вікно попередження про видалення зв’язаного з іншою таблицею у базі даних запису. Після коректної очистки таблиці програма виконує запит “delete from department”.

У вікні “Таблиця Відділ” наявне поле для введення параметру пошуку. Користувачу необхідно ввести параметр пошуку і таблиця автоматично відобразить записи, які відповідають введеному параметру. Програма в цей момент виконує алгоритм по сортуванню таблиці за допомогою класу SortedList. Алгоритм сортування можна побачити на рисунку 4.12.

```
FilteredList<Department> filteredData = new FilteredList<>(departmentsData, b -> true);
searchQuery.textProperty().addListener((observable, oldValue, newValue) -> {
    filteredData.setPredicate(department -> {
        if (newValue == null || newValue.isEmpty()) {
            return true;
        }
        String lowerCaseFilter = newValue.toLowerCase();

        if (String.valueOf(department.getId()).indexOf(lowerCaseFilter) != -1) {
            return true;
        } else if (department.getName().toLowerCase().indexOf(lowerCaseFilter) != -1) {
            return true;
        } else if (String.valueOf(department.getWorkersNumber()).indexOf(lowerCaseFilter) != -1) {
            return true;
        } else
            return false;
    });
});
SortedList<Department> sortedData = new SortedList<>(filteredData);
sortedData.comparatorProperty().bind(tableDepartments.comparatorProperty());
tableDepartments.setItems(sortedData);
```

Рисунок 4.12 Алгоритм сортування таблиці

Після успішного заповнення таблиці у вікні “Таблиця Відділ” користувач може приступити до заповнення таблиці у вікні “Таблиця Споживач”. Для заповнення таблиці користувачу необхідно ввести ідентифікатор споживача, обрати відділ у спеціальному меню, ввести назву споживача, його потужність, кількість, та кількість годин роботи цього споживача в день. Для коректної роботи з таблицею реалізовано вікна с попередженням про те що користувач заповнив не усі поля для вводу, або намагається ввести дубльований ідентифікатор, а також реалізовано вікно попередження про спробу користувачем видалити запис, який зв’язаний з іншою таблицею у базі даних. У вікні “Таблиця Споживач” також реалізовані кнопки “Додати” , “Редагувати” , “Видалити” , “Видалити все” та пошук по таблиці. Їх функції та програмна реалізація аналогічна кнопкам у вікні “Таблиця Відділ”.

4.6 Опис алгоритму формування звітів

Після успішного завершення внесення даних у таблиці Відділ і Споживач користувач має змогу передивитися звіт по енерговитратам у відділі за конкретний місяць конкретного року. Для цього користувачу потрібно перейти до вікна “Звіт енерговитрат у відділі” та вибрати в ньому необхідний відділ, місяць, ввести рік, ціну електроенергії в гривнях за 1 кВт/год. та ввести кількість відпрацьованих відділом днів. Для коректного вводу даних користувачем створено вікна с попередженням про те що користувач не заповнив усі необхідні поля. Після коректного вводу даних користувач повинен натиснути кнопку “ОК”, яка завантажує вікно зі звітом.

У вікні зі звітом програма рахує кількість відпрацьованих відділом годин, а саме програма множить введену користувачем кількість відпрацьованих відділом днів та кількість робочих годин кожного споживача у відділі, для того щоб отримати кількість робочих годин кожного споживача програма виконує запит “select from” до таблиці user у базі даних.

Також у вікні рахується сума потужностей споживачів у відділі, для цього програма виконує запит “select capacity from” до таблиці user та сумує отримані потужності.

Загальні витрати відділу рахуються програмою шляхом підсумовування витрат кожного споживача, а витрати споживача рахуються шляхом множення кількості відпрацьованих відділом годин на потужність споживача і кількість екземплярів цього споживача у відділі.

Підсумкова сума отримана шляхом множення загальних витрат відділу та ціни за електроенергію, яка була введена користувачем.

У вікні звіту також наявна діаграма, яка зображена на рисунку 4.13 і демонструє витрати кожного споживача у відділі.

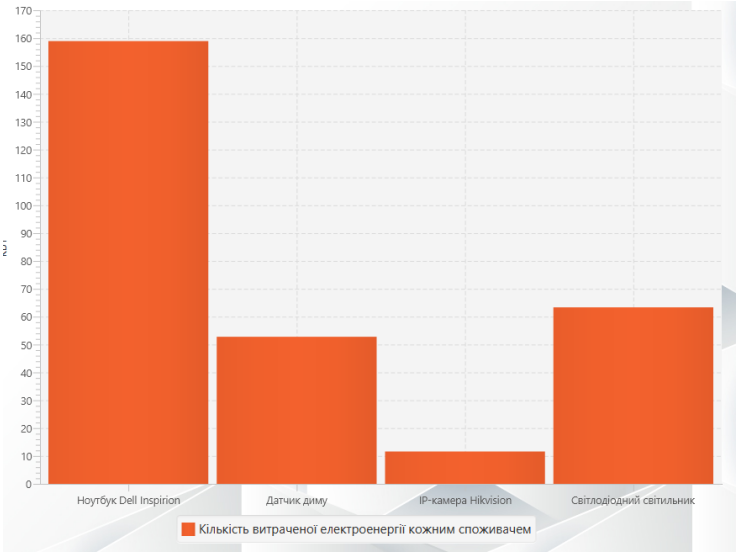

Рисунок 4.13 Діаграма витрат електроенергії кожного споживача у відділі

Всі результати у звіті програма вносить у таблицю report у базу даних за допомогою запиту “insert into”. Також у таблицю report автоматично вносяться дані про ідентифікатор звіту та ідентифікатор місяця. Ідентифікатор місяця також автоматично вноситься до таблиці month.

Загальний звіт формується із звітів за окремий місяць і рік для кожного відділу. Для отримання загального звіту користувачу необхідно спочатку сформувати звіти по кожному відділу за необхідний місяць і рік. Після формування цих звітів користувач має змогу у вікні “Загальний звіт” вибрати місяць і рік та натиснути кнопку “ОК” для отримання загального звіту підприємства. У свою чергу програма виконує запит до таблиці month “select monthId from” у базі даних для того щоб знайти звіти відділів за необхідний місяць і рік. Після цього програма у класі ShowMainReportController заповнює таблицю даними про назву відділу, загальні витрати електроенергії відділу та підсумкову суму у відділі, дані програма отримає за допомогою запиту “select from” до таблиці report у базі даних.

Також у вікні “Загальний звіт” формується кругова діаграма, яка зображена на рисунку 4.14 і демонструє кількість витраченої електроенергії кожним відділом. Дані для кругової діаграми програма збирає за допомогою запиту “select from” до таблиці report у базі даних.

Рисунок 4.14 Кругова діаграма витрат електроенергії кожного відділу

4.7 Опис алгоритму обробки виключень

Для обробки виключних ситуацій у програмному продукті було створено спеціальний клас EventHandler, в якому містяться методи, що описують порядок дій при виникненні ситуації, що не передбачена алгоритмом програми. В класі EventHandler містяться методи, що обробляють такі виключні ситуації:

— користувач не заповнив всі поля для вводу даних

— під час редагування запису користувач не ввів тексту для редагування до жодного текстового поля

— користувач не обрав поле в таблиці перед його редагуванням або видаленням

— користувач намагається створити запис у таблиці з ідентифікатором, що вже наявний у таблиці

— користувач намагається видалити запис, який пов’язаний з записом у іншій таблиці в базі даних

При обробці виключної ситуації метод класу EventHandler створює вікно з

попередженням. Приклад створеного вікна при обробці ситуації, коли користувач не заповнив всі поля для вводу даних, зображено на рисунку 4.15.

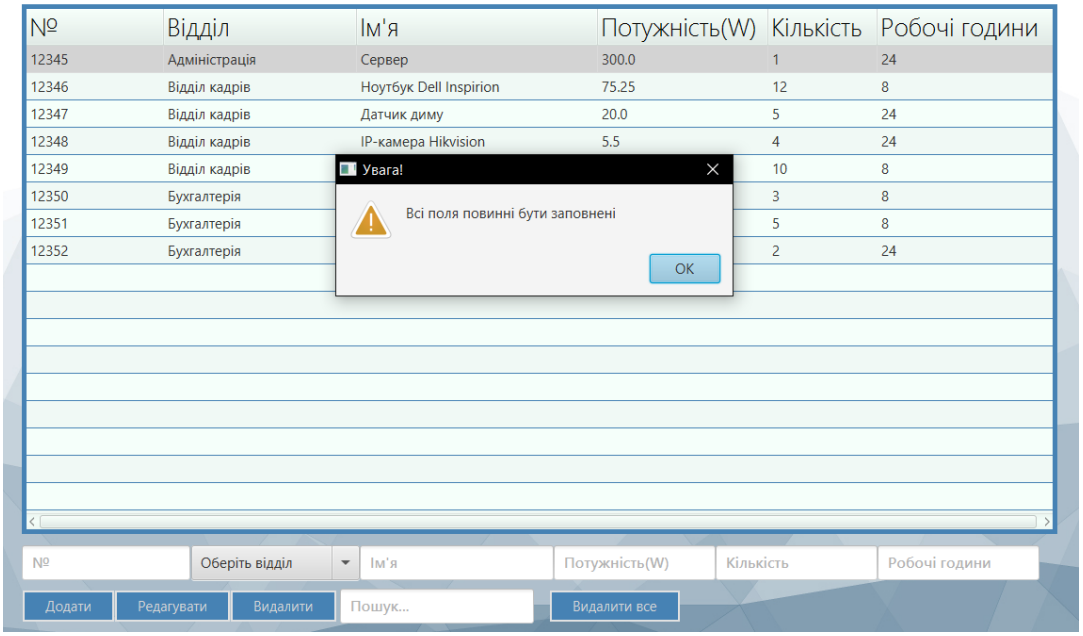

Рисунок 4.15 Діагностичне вікно для помилки не заповнення всіх полів

Для ситуації не внесення даних для одного або декількох полів при редагуванні створюється вікно з попередження, яке зображено на рисунку 4.16.

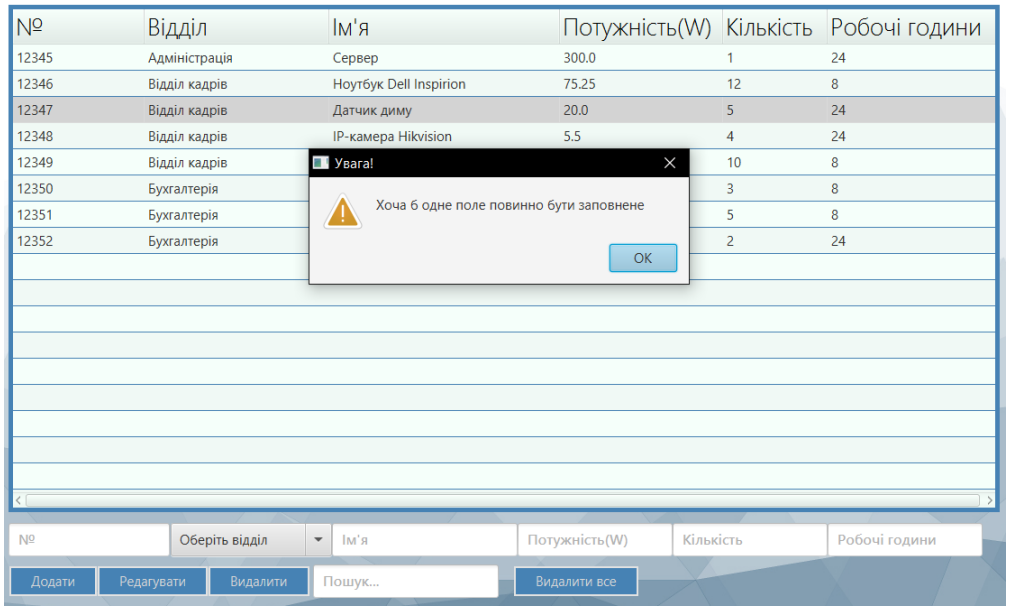

Рисунок 4.16 Діагностичне вікно для помилки не заповнення хоча б одного поля при редагуванні

Для видалення або редагування записів в таблиця користувачу необхідно спочатку вибрати запис у таблиці, при не виконанні користувачем цієї дії створюється вікно з відповідним попередженням. На рисунку 4.17 зображено відповідне вікно.

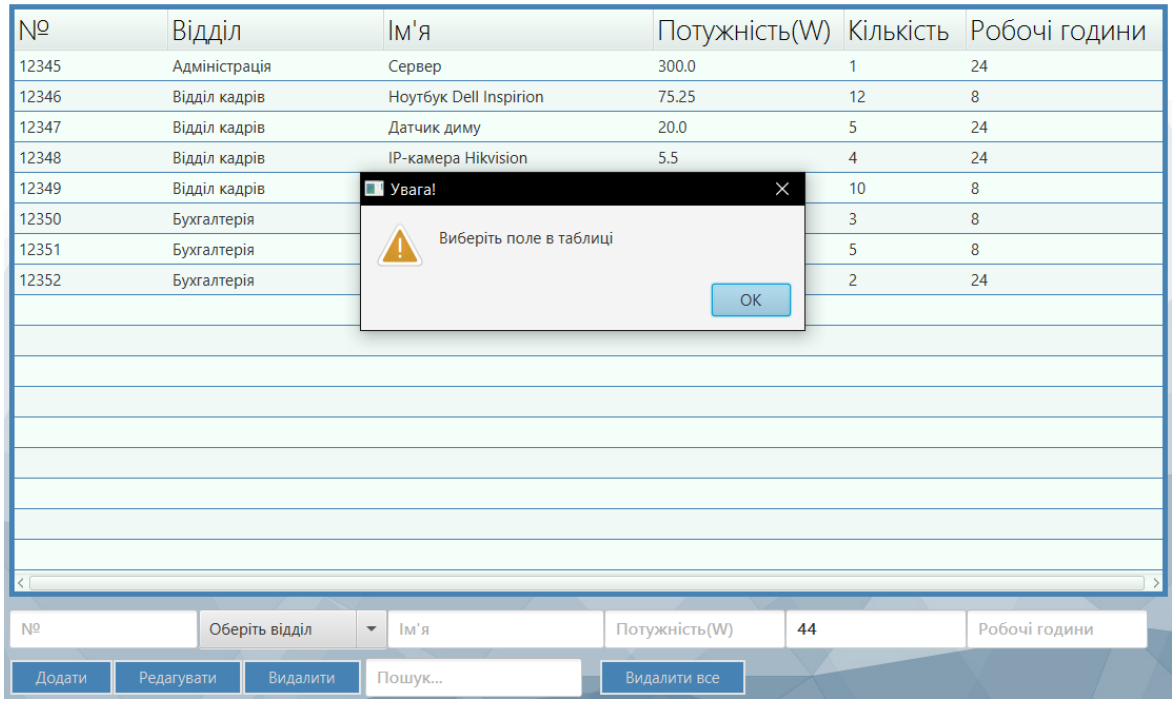

Рисунок 4.17 Діагностичне вікно для помилки не вибору поля в таблиці при редагуванні або видаленні

На рисунку 4.18 продемонстровано вікно з попередженням про спробу користувачем створення у таблиці запису с дублікатом ідентифікатора, або про спробу видалення із таблиці запису, зв’язаного з іншою таблицею.

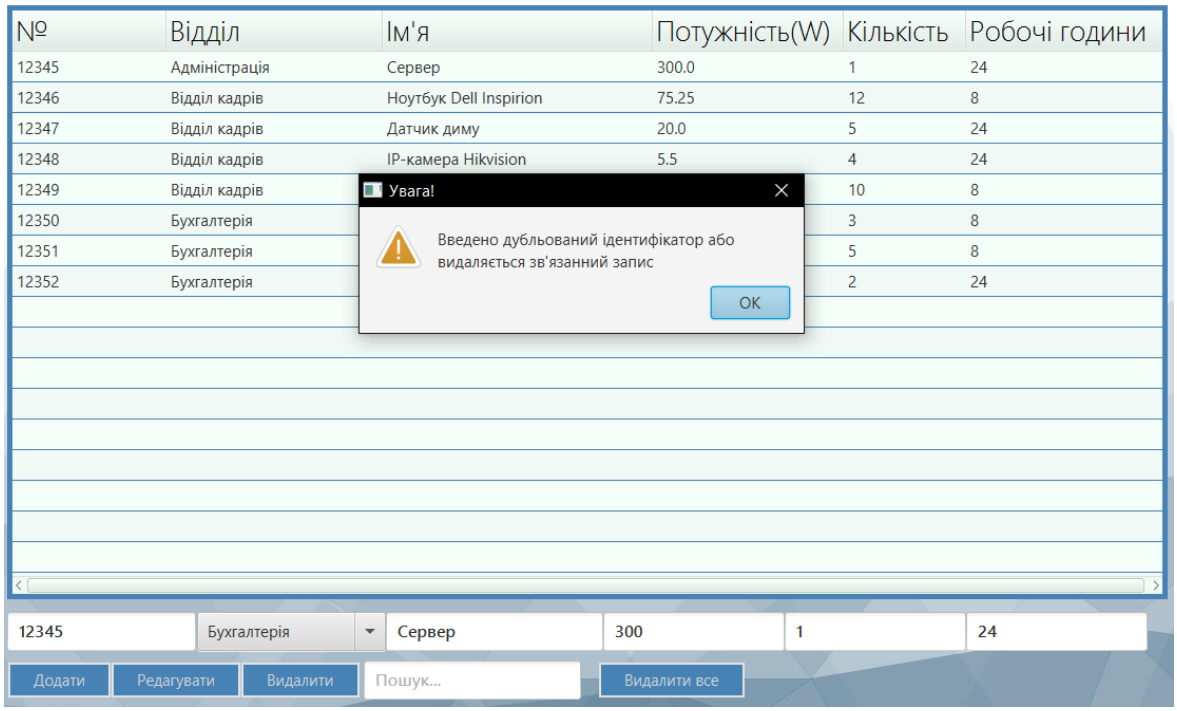

Рисунок 4.18 Діагностичне вікно для помилки введення дублікату ідентифікатора або видалення зв’язаного запису

4.8 Висновки до розділу

У розділі було описано структуру розробленого проекту, архітектуру бази даних, алгоритм взаємодії програми з базою даних, і алгоритм формування підсумкових результатів. З допомогою прикладного програмного інтерфейсу JDBC було реалізовано швидкий і зручний доступ до бази даних, завдяки якому користувач розробленої системи має змогу використовувати зрозумілу навігацію для взаємодії з базою даних, та швидко отримувати детальні підсумкові результати у звітах.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

5.1. Створення бази даних на сервері

Для розгортання бази даних на сервері необхідно створити локальне з’єднання із сервером в інтерфейсі MySQL Workbench та виконати програмний код, файл із яким, надається разом із програмним продуктом.

Для створення локального з’єднання, необхідно на головній сторінці MySQL Workbench натиснути на символ додавання (рисунок 5.1).

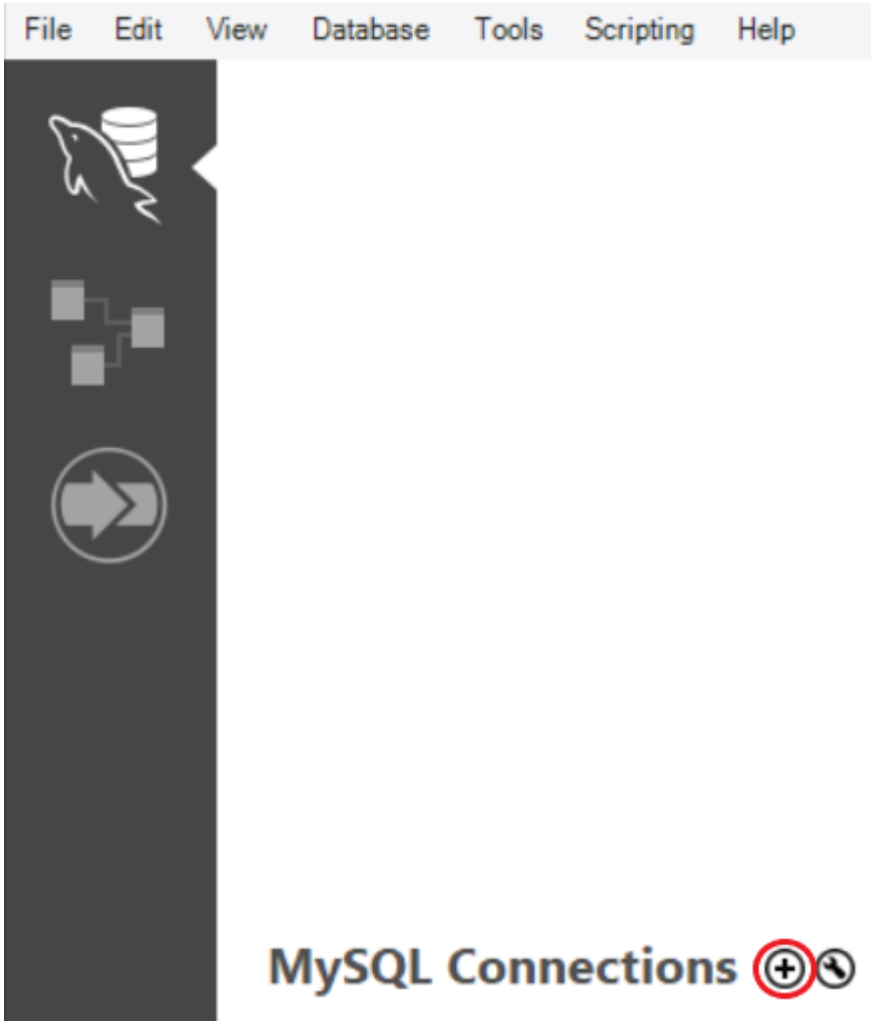

Рисунок 5.1 Створення нового підключення до серверу

У вікні налаштувань (рисунок 5.2) необхідно заповнити поле “Connection Name” назвою створюваного з’єднання, вказати id-адресу сервера (тут 127.0.0.1), порт протоколу TCP/IP (тут 3306), по якому буде встановлюватися зв’язок, а також вказати ім’я користувача (тут root).

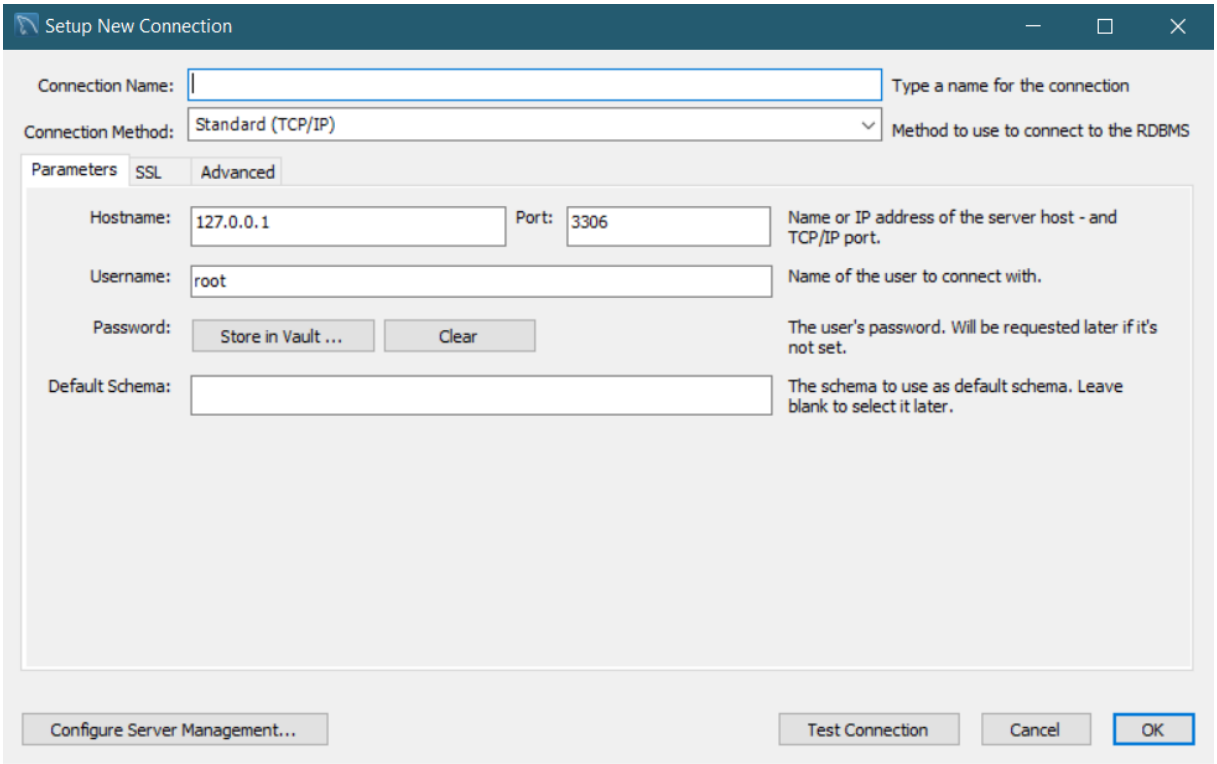

Рисунок 5.2 Вікно налаштування

Для перевірки можливості створення з’єднання необхідно натиснути на кнопку “Test Connection”, та ввести пароль доступу до сервера у вікні авторизації (рисунок 5.3).

Рисунок 5.3 Вікно авторизації користувача

Якщо всі дані введені вірно та тестове підключення виконано успішно, про що користувач буде повідомлений діагностичним вікном (рисунок 5.4) потрібно натиснути кнопку “OK” у вікні налаштування з’єднання.

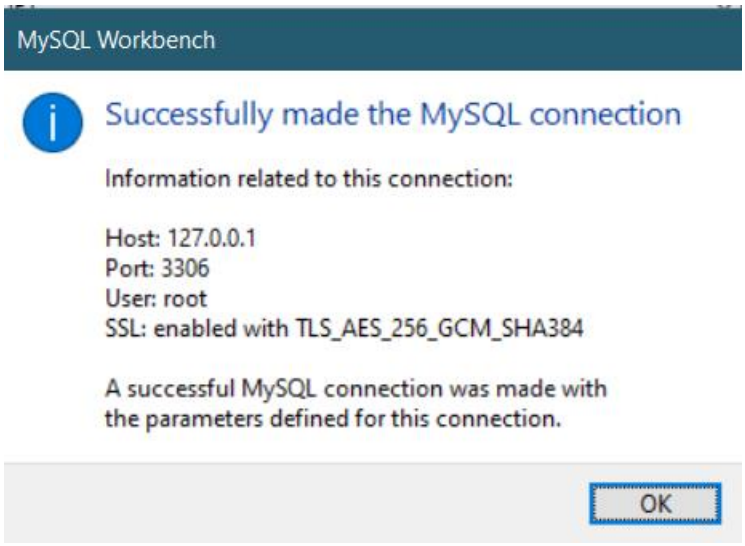

Рисунок 5.4 Діагностичне вікно коректного тестового підключення

Після створення нового з’єднання, воно відобразиться у стартовому вікні MySQL Workbench (рисунок 5.5).

Рисунок 5.5 Новий зв’язок на головній сторінці MySQL Workbench

Подвійне натискання на нове з’єднання викличе вікно авторизації (рисунок 5.3) і після коректного введення паролю буде відкрите вікно текстового менеджера бази даних (рисунок 5.6).

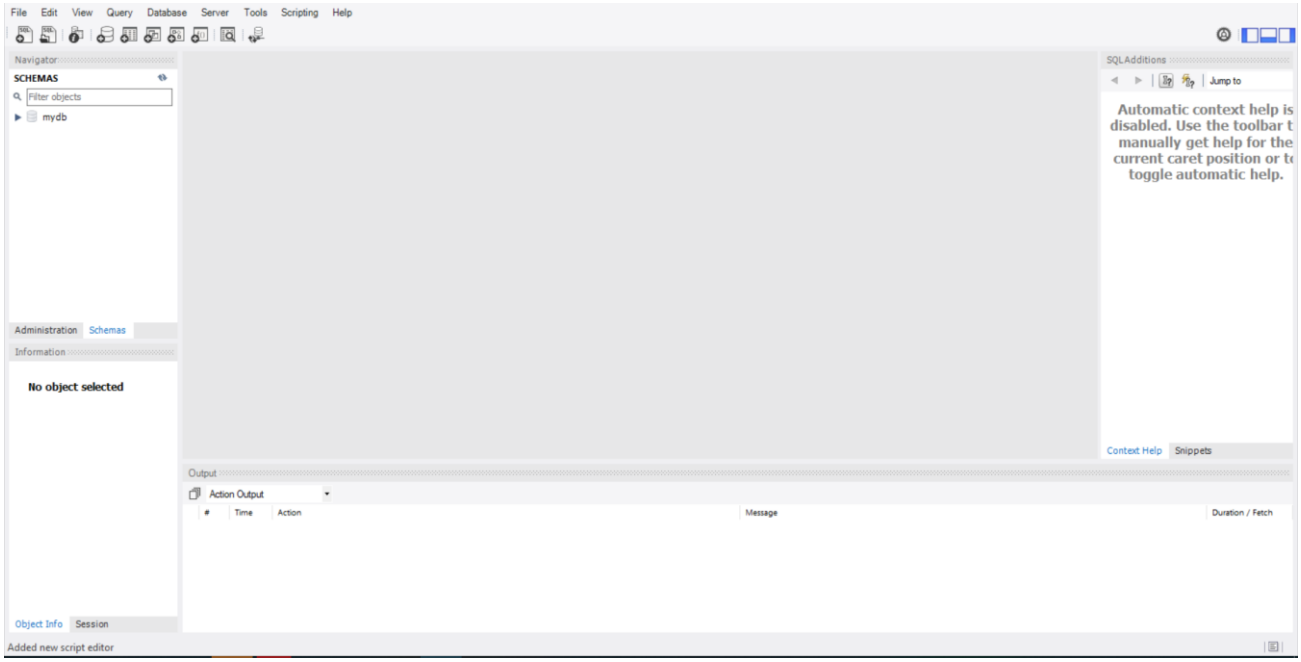

Рисунок 5.6. Вікно текстового менеджера бази даних

Натиснувши File – Open SQL Script, після чого обравши файл конфігурації бази даних необхідно натиснути символ блискавки у вікні, що відкрилося (рисунок 5.7) та дочекатися закінчення виконання програми розгортання бази даних.

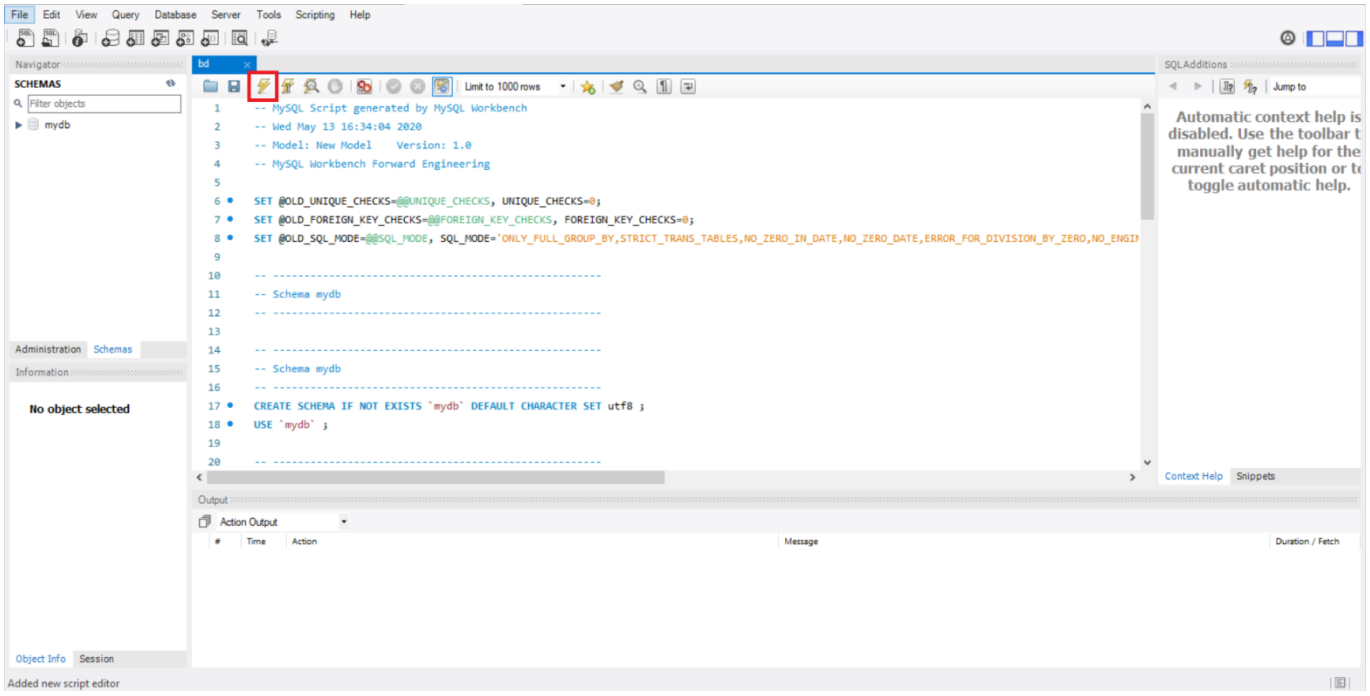

Рисунок 5.7. Запуск розгортання бази даних на сервері

Після закінчення завантаження, база даних розгорнута і готова до використання та обробки.

У випадку необхідності ручного адміністрування бази даних поза функціоналом створеної програмної системи необхідно створити новий файл типу “SQL Text File" обравши пункт “New Query Tab” в меню “File” (рисунок 5.8).

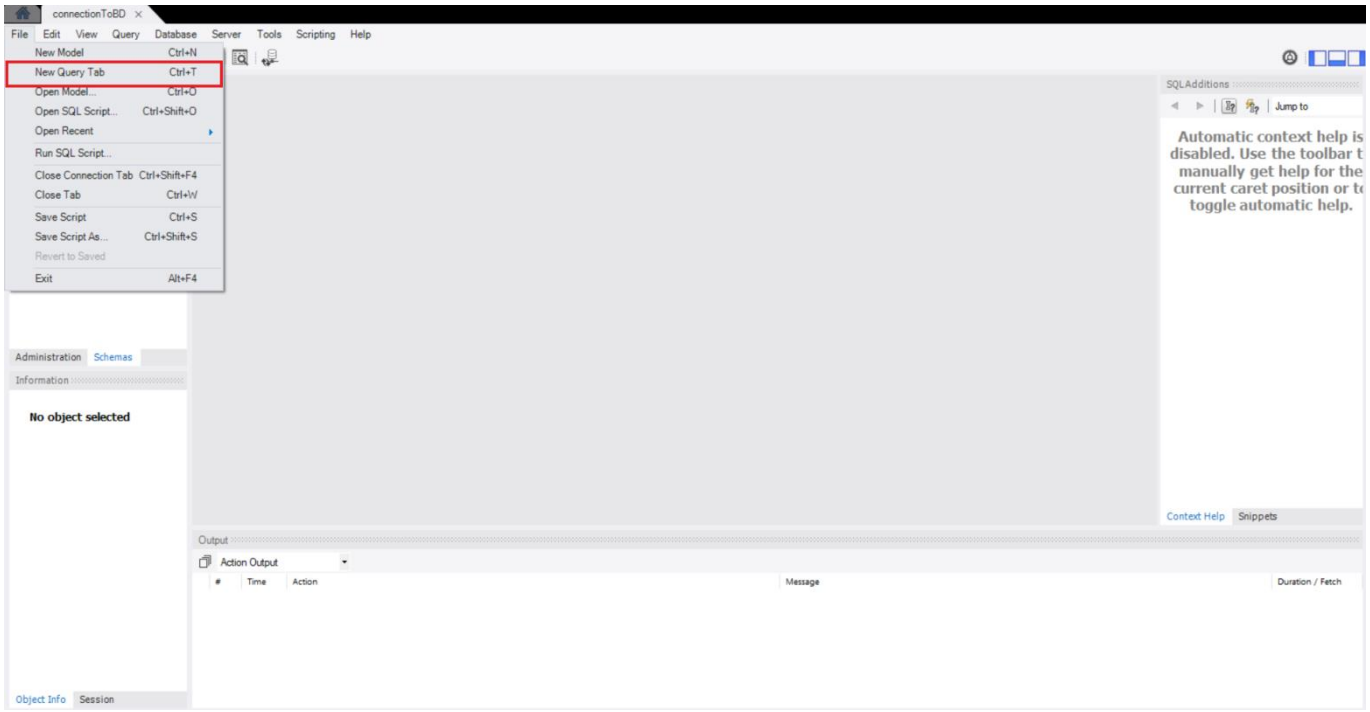

Рисунок 5.8. Створення нового файлу SQL-сценарію

Перед запуском даного файлу із створеним в ньому сценарієм в обробку базою даних, створений файл необхідно зберегти, через натискання комбінації клавіш “Ctrl + s” або через вибір пункту “Save Script” в меню “File”. Далі необхідно обрати місце зберігання нового файлу (рисунок 5.9) в системі комп’ютера користувача та підтвердити збереження.

Проте, хоч функціонал менеджменту бази даних в ручному режимі та в обхід алгоритмів програмної системи і існує, втручання в роботу бази даних може призвести до некоректної роботи програмної системи, або взагалі до неможливості користування програмною системою. Тому рекомендовано використовувати функціонал MySQL Workbench із можливістю редагування бази даних в ручному

режимі лише для цілей корегування даних, помилково внесених користувачем та налаштування оточення серверу бази даних.

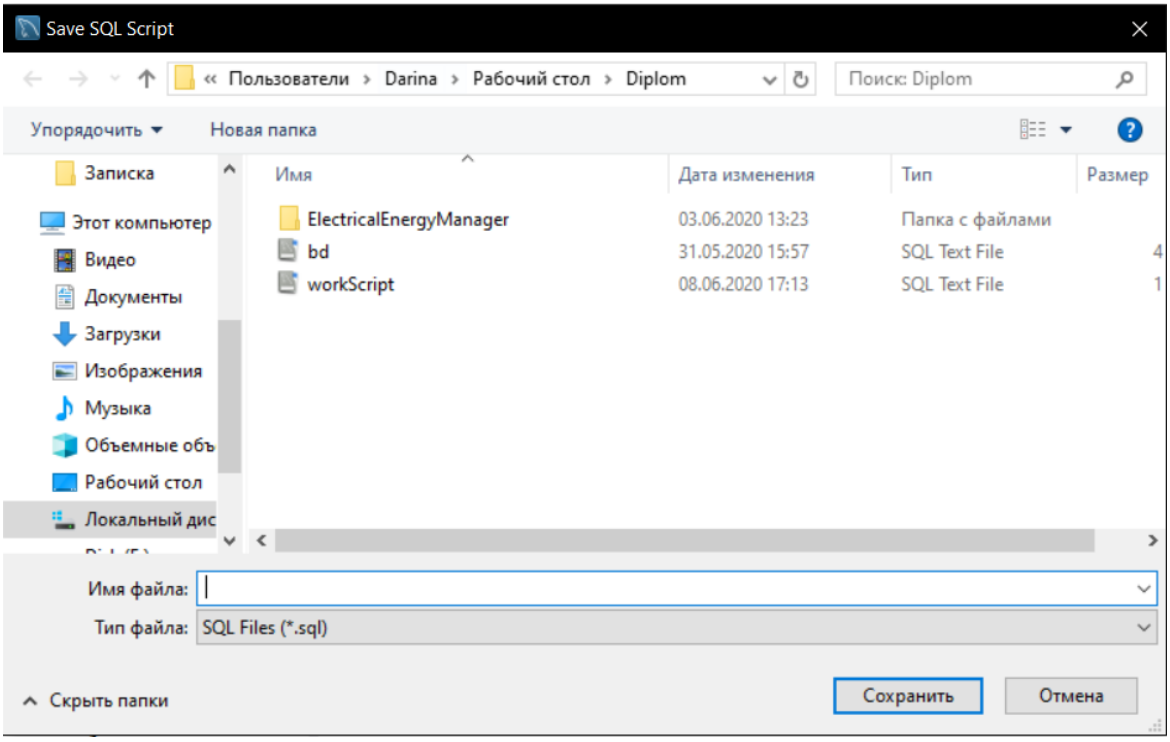

Рисунок 5.9 Збереження файлу SQL-сценарія

Також у функціоналі середовища MySQL передбачена можливість вимкнення робочого серверу бази даних задля вивільнення апаратних ресурсів комп'ютера в час коли програмна система та сервер бази даних не експлуатуються.

Для закриття серверу необхідно обрати із панелі задач MySQL Notifier (рисунок 5.10), викликати на ньому контекстне меню, та обрати пункт "Stop". Після підтвердження зупинки серверу та перехід його із стану "Running" в режим "Stopped" сервер буде зупинено.

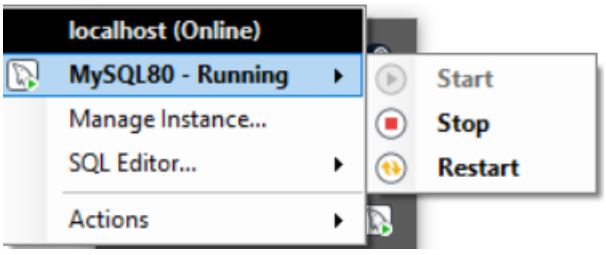

Рисунок 5.10. Функціонал адміністрування серверу через MySQL Notifier

У випадку потреби запуску серверу знов, необхідно запустити MySQL Notifier, обрати на ньому пункт “Start” і дочекатися переходу серверу в стан “Running”. Після запуску серверу необхідно знов встановити з’єднання із ним.

Після завершення всіх вище описаних дій та встановлення з’єднання із сервером, він вважається готовим до роботи і може використовуватися програмною системою.

5.2 Робота з віконним додатком

Користувач запускає додаток формату .jar і заповнює таблиці Відділ і Споживач. Для внесення запису в таблицю Відділ користувачу необхідно заповнити всі поля для вводу як зображено на рисунку 5.11. Після заповнення полів користувач повинен натиснути кнопку “Додати”, щоб додати запис до таблиці.

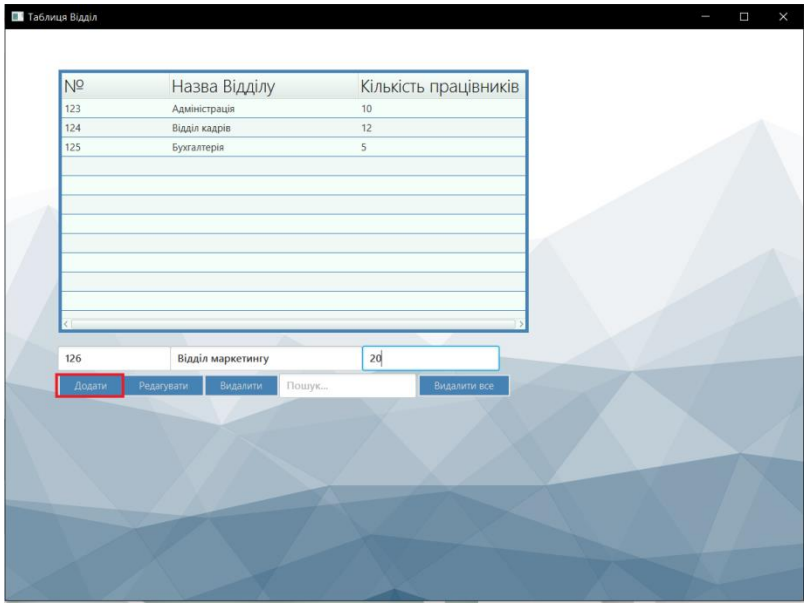

Рисунок 5.11 Додавання запису до таблиці Відділ

Для внесення запису в таблицю Споживач користувачу необхідно заповнити всі поля для вводу та вибрати в меню потрібний Відділ як зображено на рисунку

5.12. Після заповнення полів користувач повинен натиснути кнопку “Додати”, щоб додати запис до таблиці.

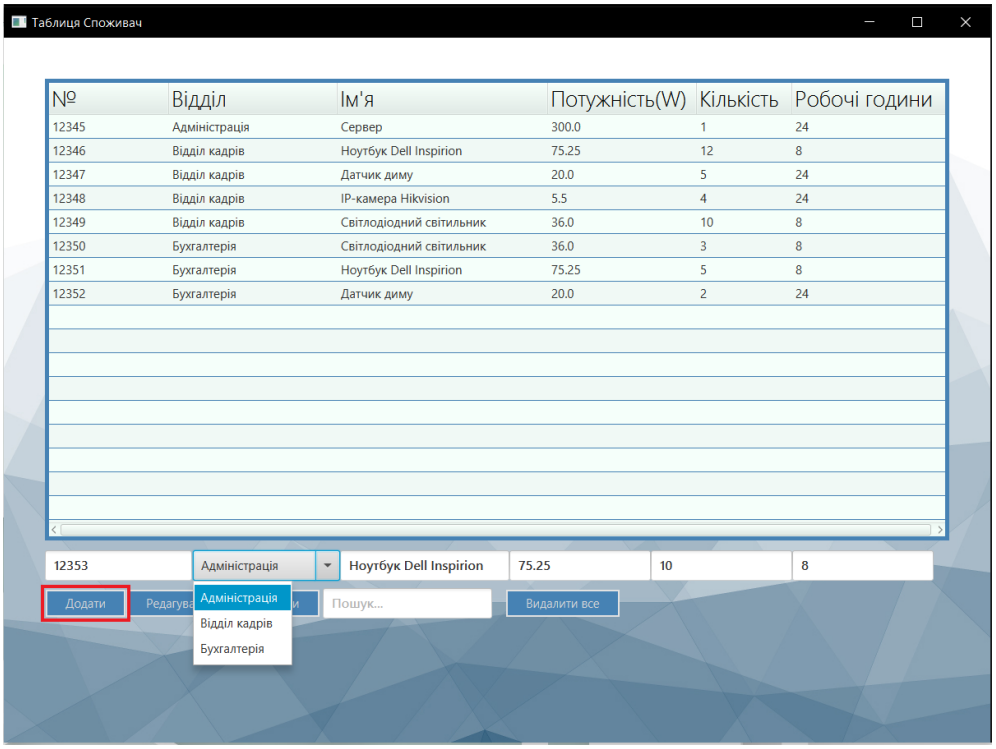

Рисунок 5.12 Додавання запису до таблиці Споживач

Для видалення або редагування вже існуючого в таблиці запису користувачу потрібно вибрати в таблиці необхідний запис і натиснути кнопку “Видалити” або ввести в поле/поля, в яких користувачу потрібно зробити зміну, необхідну інформацію та натиснути кнопку “Редагувати”.

Для пошуку по таблиці користувачу необхідно ввести параметр пошуку в поле пошуку і таблиця автоматично відсортує записи по заданому параметру.

Для очищення таблиці користувачу потрібно натиснути кнопку “Видалити все”.

Для перегляду звіту енерговитрат у відділі користувач повинен вибрати у меню пункт “Звіт енерговитрат у відділі” і у вікні вибрати відділ, місяць, ввести рік, ціну за електроенергію у гривнях за 1 кВт/год., та ввести кількість відпрацьованих відділом годин як зображено на рисунку 5.13.

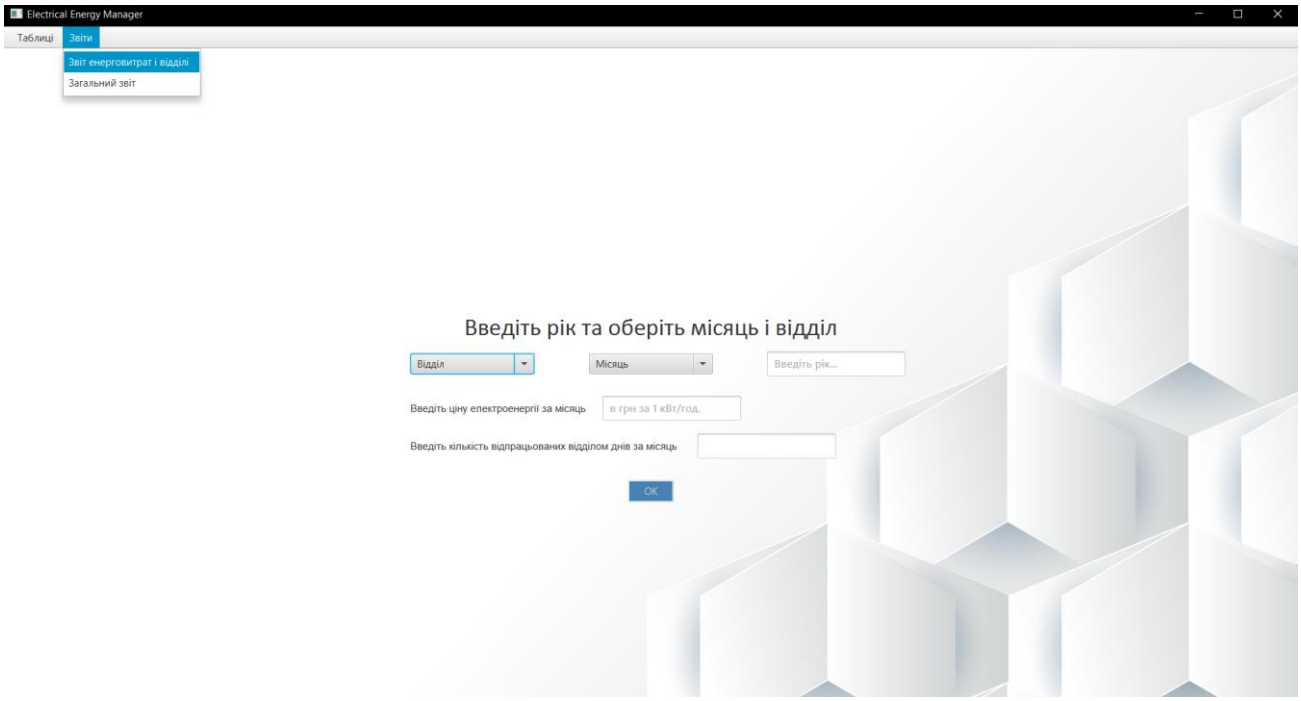

Рисунок 5.13 — Вікно вибору даних для звіту енерговитрат у відділі

Для перегляду загального звіту по всьому підприємству користувач повинен вибрати у меню пункт “Загальний звіт” і у вікні вибрати місяць і рік, як зображено на рисунку 5.14.

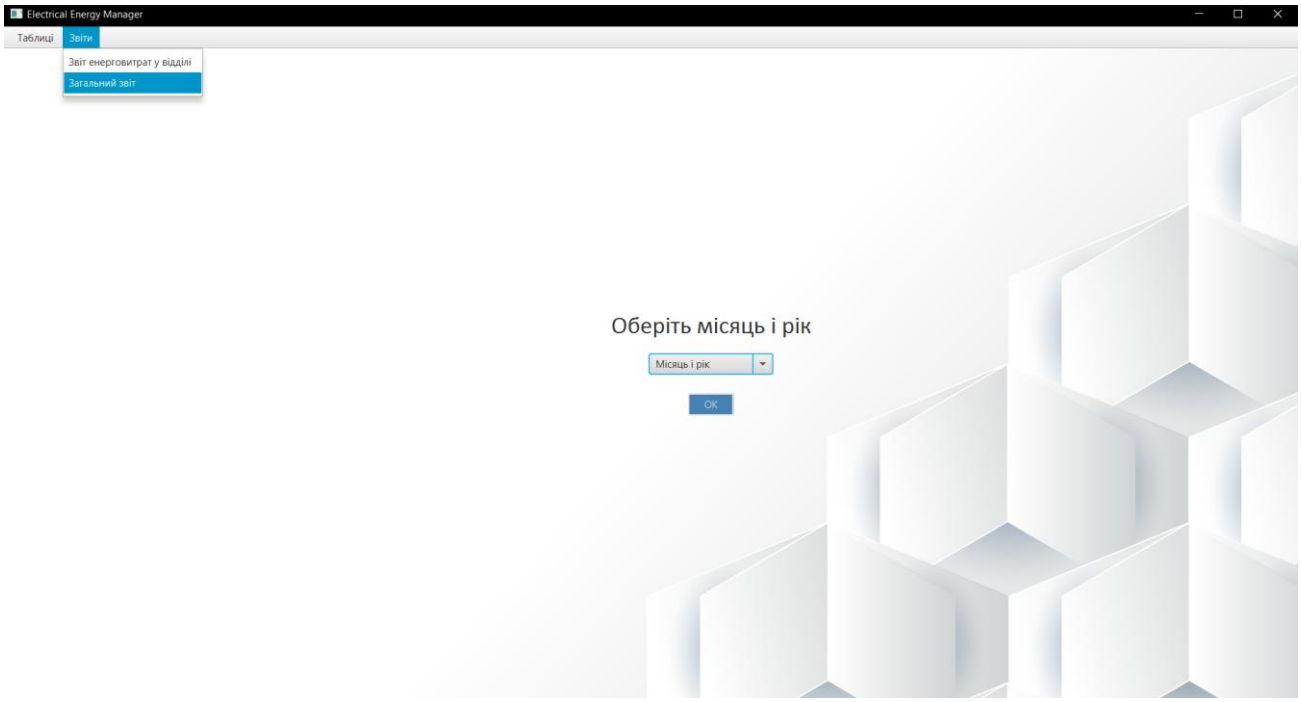

Рисунок 5.14 — Вікно вибору даних для загального звіту енерговитрат

5.3. Системні вимоги

Для роботи з розробленою програмною системою та сервером SQL користувачу потрібні мінімальні потужності апаратного забезпечення. Вимоги наведені в таблиці 5.1.

Пристрій	Характеристика
Процесор	x64 з тактовою частотою не нижче 1.4 GHz. Рекомендована тактова частота 2.0 або краще.
Тип процесора	AMD Opteron/ AMD Athlon 64/ Intel Xeon/ Intel Pentium IV.
Оперативна пам’ять	Мінімум 1 ГБ RAM. Рекомендовано не менше 4 ГБ RAM.
Монітор	Super-VGA (800x600) або краще.
Жорсткий диск	Не менше 10 ГБ вільного місця
Операційна система	Windows 10

Таблиця 5.1. Вимоги до апаратного забезпечення користувача

Підтримувані апаратні архітектури:

— 32-розрядна (x86)

— 64-розрядна (x64).

5.4 Висновки до розділу

У розділі була проведена презентація роботи користувача з програмою. Програма має надавати такі можливості:

- внесення даних у таблиці в базі даних
- редагування і видалення даних в таблицях
- аналіз кількості витраченої електроенергії та грошових витрат на електроенергію у відділі
- формування на основі інформації у базі даних, і отриманих програмою підрахунків звіту енерговитрат у відділі за місяць
- формування на основі створених звітів загального звіту витраченої електроенергії у всіх відділах

ВИСНОВКИ

В ході виконання дипломної роботи було досліджено предметну область енергоменеджменту на підприємстві, а також проаналізовані наявні системи, що вирішують задачу автоматизації енергоменеджменту.

За результатами дослідження було виявлено, що наявні системи автоматичного енергоменеджменту, хоч і є достатньо автономними та об’ємними, проте є також і достатньо громіздкими, дорогими та складними в експлуатації.

На основі проведених досліджень була створена програмна система енергоменеджменту на підприємстві, яка є доволі актуальною, так як вирішує проблему енергоменеджменту, в першу чергу, зі сторони зручності користування системою.

Розроблений програмний продукт дозволяє здійснювати контроль, аналіз та підрахунок витрат електроенергії на підприємстві. Також передбачена можливість формування звітів для окремого відділу та для кожного місяця.

Актуальність створеного програмного продукту полягає в простоті його використання, невисоких вимог до апаратної частини комп’ютеру користувача, та наявності візуальних елементів аналізу, представлених у вигляді діаграм, що демонструють використання електроенергії на підприємстві.

Користувачами даної програмної системи можуть бути різні організації та підприємства, що бажають здійснювати контроль та аналіз енергоспоживання своєї структури. Завдяки універсальності системи вона може використовуватися як доволі великими компаніями, так і підприємствами малого бізнесу.

Проведенням серії тестувань створеного програмного забезпечення була підтверджена коректність роботи програмної системи, коректність синхронізації даних між системою та базою даних, а також простота та зрозумілість користувацького інтерфейсу, а отже створена програмна система відповідає поставленим вимогам розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ExpertPower [Електронний ресурс] — Режим доступу до ресурсу: http://www.satec-global.com.ua/uk/programne-zabezpechennya/expertpower/

2. ISO 50001 Energy management [Електронний ресурс] — Режим доступу до ресурсу: https://www.iso.org/iso-50001-energy-management.html.

3. ISO 50001: 2011 Energy management systems — Requirements with guidance for use [Електронний ресурс] — Режим доступу до ресурсу: https://www.iso.org/standard/51297.html.

4. Герберт Шилдт. Java. Полное руководство, 10-е. — М.: Диалектика, 2018. — 1488 с.

5. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL: полное руководство, 2014. — 960 с.

6. JDBC Introduction [Електронний ресурс] — Режим доступу до ресурсу: https://docs.oracle.com/javase/tutorial/jdbc/overview/index.html.

7. What is Maven? [Електронний ресурс] — Режим доступу до ресурсу: https://maven.apache.org/what-is-maven.html.

8. JavaFX Overview [Електронний ресурс] — Режим доступу до ресурсу: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.html.

9. InteliJ IDEA overview [Електронний ресурс] — Режим доступу до ресурсу: https://www.jetbrains.com/help/idea/discover-intellij-idea.html.

10. Cascading Style Sheets home page [Електронний ресурс] — Режим доступу до ресурсу: https://www.w3.org/Style/CSS/Overview.en.html.

Додаток 1

Розробка програмного забезпечення для автоматизації енергоменеджменту на підприємстві на платформі ОС Windows

Специфікація

УКР.НТУУ“КПІ”.ТР61118_20Б

Аркушів 2

2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ“КПІ”.ТР61118_20Б 81-1	Записка	Пояснювальна записка
Компоненти		
УКР.НТУУ“КПІ”.ТР61118_20Б 12-1	Текст програмного модулю	
УКР.НТУУ“КПІ”.ТР61118_20Б 13-1	Опис програмного модулю	

Додаток 2

Розробка програмного забезпечення для автоматизації енергоменеджменту на підприємстві на платформі ОС Windows

Текст програмного модулю

УКР.НТУУ“КПІ”.ТР61118_20Б 12-1

Аркушів 10

2020

```
package Controllers;
import Connectors.Connector;
import Events.EventHandler;
import Tables.Report;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.Scene;
import javafx.scene.chart.*;
import javafx.scene.control.Label;
import javafx.stage.Stage;
import javafx.scene.Parent;
import javafx.fxml.FXMLLoader;
import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class ShowDepartmentReportController {

    @FXML
    private Label monthNumberOutput;

    @FXML
    private Label yearOutput;

    @FXML
    private Label priceEnterOutput;

    @FXML
    private Label departmentNameOutput;

    @FXML
    private Label workedHoursOutput;

    @FXML
    private Label capacitySumOutput;

    @FXML
    private Label consumptionOutput;

    @FXML
    private Label expenditureSumOutput;

    @FXML
    CategoryAxis xAxis = new CategoryAxis();

    @FXML
    NumberAxis yAxis = new NumberAxis();

    @FXML
    StackedBarChart<String, Number> usersDiagram = new StackedBarChart<String,
Number>(xAxis, yAxis);

    XYChart.Series<String, Number> series1 = new XYChart.Series<String, Number>();

    ObservableList<XYChart.Series<String, Number>> userDiagramData =
FXCollections.observableArrayList();
```

```
    ArrayList <Report> reports;

    Connection connection = null;

    Statement statement = null;

    ResultSet resultSet = null;

    ArrayList <String> usName = new ArrayList<>();
    ArrayList <Double> usCapacity = new ArrayList<>();
    ArrayList <Integer> usNumber = new ArrayList<>();
    ArrayList <Integer> usWorkHours = new ArrayList<>();

    int capSum = 0;

    int id = 0;

    @FXML
    public void initialize() {
        usersDiagram.setAnimated(true);
        reports = new ArrayList<>();
        connection = Connector.connection;
        String depName = "";
        double capacitySum = 0;
        try {
            statement = connection.createStatement();
            resultSet = statement.executeQuery("select departmentName from department where
departmentId = " + departmentReportController.departmentId);
            while (resultSet.next()) {
                depName = resultSet.getString(1);
            }
            resultSet = statement.executeQuery("select capacity, userNumber from user where
departmentId = " + departmentReportController.departmentId);
            while (resultSet.next()) {
                capacitySum += Double.parseDouble(resultSet.getString(1)) *
resultSet.getInt(2);
            }
            resultSet = statement.executeQuery("select userName, capacity, userNumber,
workHours from user where departmentId = " + departmentReportController.departmentId);
            while (resultSet.next()) {
                usName.add(resultSet.getString(1));
                usCapacity.add(Double.parseDouble(resultSet.getString(2)));
                usNumber.add(Integer.parseInt(resultSet.getString(3)));
                usWorkHours.add(resultSet.getInt(4));
            }
        } catch (SQLException e) {
            EventHandler.SQLEx();
            return;
        }
        updateChart();
        int hours = 0;
        for (int i = 0; i < usWorkHours.size(); i++) {
            hours += usWorkHours.get(i);
        }
        monthNumberOutput.setText(String.valueOf(departmentReportController.monthNumber));
        yearOutput.setText(String.valueOf(departmentReportController.year));
        priceEnterOutput.setText(String.valueOf(departmentReportController.price));
        departmentNameOutput.setText(depName);
        workedHoursOutput.setText(String.valueOf(departmentReportController.workedHours *
hours));
```

```
        capacitySumOutput.setText(String.valueOf(capacitySum / 1000));
        consumptionOutput.setText(String.valueOf(capSum));
        expenditureSumOutput.setText(String.valueOf((capSum *
departmentReportController.price)));
        double expSum = (capSum * departmentReportController.price);
        double workHours = departmentReportController.workedHours * hours;
        try {
            resultSet = statement.executeQuery("select reportId from report");
            while (resultSet.next()) {
                id = resultSet.getInt(1);
            }
        } catch (SQLException e) {
            EventHandler.SQLEx();
            return;
        }
        id++;
        try {
            statement.executeUpdate("insert into report (reportId, consumption,
departmentId, monthNumber, capacitySum, yearNumber, workedHours, monthId, expenditureSum,
departmentName, price) values (" + id + ", " + capSum + ", " +
departmentReportController.departmentId + ", " + departmentReportController.monthNumber + ",
" + capacitySum / 1000 + ", " + departmentReportController.year + ", " + workHours + ", " +
departmentReportController.monthId + ", " + expSum + ", '" + depName + "', " +
departmentReportController.price + ")");
        } catch (SQLException e) {
        }
        try {
            statement.executeUpdate("insert into month (monthId) values (" +
departmentReportController.monthId + ")");
        } catch (SQLException e) {
        }
    }

private void updateChart() {
        xAxis.setCategories(FXCollections.<String>observableArrayList());
        usersDiagram.getData().clear();
        series1.setName("Кількість витраченої електроенергії кожним споживачем");
        for (int i = 0; i < usName.size(); i++) {
            capSum += (usCapacity.get(i) * usNumber.get(i) *
departmentReportController.workedHours * usWorkHours.get(i)) / 1000;
            series1.getData().add(new XYChart.Data<String, Number>(usName.get(i),
(usCapacity.get(i) * usNumber.get(i) * departmentReportController.workedHours *
usWorkHours.get(i)) / 1000));
        }
        yAxis.setLabel("кВт");
        xAxis.setCategories(FXCollections.<String>observableArrayList(usName));
        usersDiagram.getData().addAll(series1);
    }
    protected void UserController(Stage stage) throws IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("/FXML/tableViews/userView.fxml"));
        Scene scene = new Scene(root, 1200, 860);
        stage.setTitle("Таблиця Споживач");
        stage.setScene(scene);
        stage.show();
    }

    @FXML
    public void departmentTable(javafx.event.ActionEvent event) throws IOException {
        Stage stage = new Stage();
        departmentController(stage);
```

```
    }

    protected void departmentController(Stage stage) throws IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("/FXML/tableViews/departmentView.fxml"));
        Scene scene = new Scene(root, 1200, 860);
        stage.setTitle("Таблиця Відділ");
        stage.setScene(scene);
        stage.show();
    }

    public void mainReportEnterData(ActionEvent event) throws IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("/FXML/reportViews/mainReportEnterDataView.fxml"));
        StageContainer.stage.setScene(new Scene(root, 1920, 1000));
        StageContainer.stage.show();
    }

    public void departmentReportEnterData(ActionEvent event) throws IOException {
        Parent root =
FXMLLoader.load(getClass().getResource("/FXML/reportViews/departmentReportEnterDataView.fxml
"));
        StageContainer.stage.setScene(new Scene(root, 1920, 1000));
        StageContainer.stage.show();
    }

}

public class UserController {

    private ObservableList usersData = FXCollections.observableArrayList();

    @FXML
    private TableView<User> tableUsers;

    @FXML
    private TableColumn<User, Integer> idColumn;

    @FXML
    private TableColumn<User, String> depIdColumn;

    @FXML
    private TableColumn<User, String> nameColumn;

    @FXML
    private TableColumn<User, Double> capacityColumn;

    @FXML
    private TableColumn<User, Integer> numberColumn;

    @FXML
    private TableColumn<User, Integer> workHoursColumn;

    @FXML
    private TextField textId;

    @FXML
    private SplitMenuButton departmentMenuButton;

    @FXML
    private TextField textName;
```

```
    @FXML
    private TextField textCapacity;

    @FXML
    private TextField textNumber;

    @FXML
    private TextField textWorkHours;

    @FXML
    private TextField searchQuery;


    public ResultSet resultSet;

    private Connection connection;

    Statement statement = null;

    ResultSet rs = null;
    @FXML
    private void initialize() {
        tableUsers.getSelectionModel().setSelectionMode(
                SelectionMode.MULTIPLE
        );
        connection = Connector.connection;
        resultSet = null;
        usersData.clear();
        try {
            statement =  connection.createStatement();
            resultSet = statement.executeQuery("select * from user");
        }
        catch (SQLException e) {
            EventHandler.SQLEx();
            return;
        }
        initData();
        idColumn.setCellValueFactory(new PropertyValueFactory<User, Integer>("id"));
        depIdColumn.setCellValueFactory(new PropertyValueFactory<User,
String>("departmentName"));
        nameColumn.setCellValueFactory(new PropertyValueFactory<User, String>("name"));
        capacityColumn.setCellValueFactory(new PropertyValueFactory<User,
Double>("capacity"));
        numberColumn.setCellValueFactory(new PropertyValueFactory<User,
Integer>("userNumber"));
        workHoursColumn.setCellValueFactory(new PropertyValueFactory<User,
Integer>("workHours"));
        tableUsers.setItems(usersData);
        FilteredList<User> filteredData = new FilteredList<>(usersData, b -> true);
        searchQuery.textProperty().addListener((observable, oldValue, newValue) -> {
            filteredData.setPredicate(user -> {

                if (newValue == null || newValue.isEmpty()) {
                    return true;
                }

                String lowerCaseFilter = newValue.toLowerCase();

                if (String.valueOf(user.getId()).indexOf(lowerCaseFilter) != -1) {
                    return true;
```

```
                } else if (user.getName().toLowerCase().indexOf(lowerCaseFilter) != -1) {
                    return true;
                } else if (String.valueOf(user.getCapacity()).indexOf(lowerCaseFilter) != -
1) {
                    return true;
                } else if (String.valueOf(user.getDepartmentId()).indexOf(lowerCaseFilter)
!= -1) {
                    return true;
                } else if (String.valueOf(user.getUserNumber()).indexOf(lowerCaseFilter)!=-
1) {
                    return true;
                } else if (String.valueOf(user.getWorkHours()).indexOf(lowerCaseFilter) != -
1) {
                    return true;
                } else
                    return false;
            });
        });

        SortedList<User> sortedData = new SortedList<>(filteredData);
        sortedData.comparatorProperty().bind(tableUsers.comparatorProperty());
        tableUsers.setItems(sortedData);
        departmentMenuButton.getItems().clear();
        try {
            resultSet = statement.executeQuery("select * from department");
            while (resultSet.next()) {
                String depName = resultSet.getString(2);
                MenuItem choice = new MenuItem(depName);
                choice.setOnAction((e)-> {
                    try {
                        rs = statement.executeQuery("select departmentId from department
where departmentName = '" + depName + "'");
                        while (rs.next()) {
                            departmentMenuButton.setText(depName);
                            departmentId = rs.getInt(1);
                        }
                    } catch (SQLException e1) {
                        EventHandler.SQLEx();
                        return;
                    }
                });
                departmentMenuButton.getItems().addAll(choice);
            }
        } catch (SQLException e) {
            EventHandler.SQLEx();
            return;
        }
    }

    private void initData() {
        try {
            Statement stm = connection.createStatement();
            ResultSet r = null;
            while (resultSet.next()) {
                int id = resultSet.getInt(1);
                String name = resultSet.getString(2);
                double capacity = Double.parseDouble(resultSet.getString(3));
                int depId = resultSet.getInt(5);
                int number = resultSet.getInt(4);
                int hours = resultSet.getInt(6);
                String departmentName = "";
```

```
                r = stm.executeQuery("select departmentName from department where
departmentId = " + depId);
                while (r.next()) {
                    departmentName = r.getString(1);
                    System.out.println(departmentName);
                }
                User u = new User(id, name, capacity,depId, departmentName, number, hours);
                usersData.add(u);
            }
        } catch (SQLException e) {
            EventHandler.SQLEx();
            return;
        }
    }

    private int departmentId = 1;
    @FXML
    public void Add(ActionEvent event) {
        boolean isInsert = true;
        int userId = 1;
        String userName = "";
        double userCapacity = -1.0;
        int userNumber = 1;
        int userWorkHours = 1;
        try {
            userId = Integer.parseInt(textId.getText());
            userName = textName.getText();
            userCapacity = Double.parseDouble(textCapacity.getText());
            userNumber = Integer.parseInt(textNumber.getText());
            userWorkHours = Integer.parseInt(textWorkHours.getText());
        } catch (NumberFormatException e) {
            EventHandler.allEmptyField();
            return;
        }
        if (isInsert == true) {
                try {
                    statement.executeUpdate("insert into user (userId, userName, capacity,
departmentId, userNumber, workHours) values (" + userId + ", '" + userName + "', " +
userCapacity + ", " + departmentId + ", " + userNumber + ", " + userWorkHours + ")");
                } catch (SQLException e1) {
                    EventHandler.SQLEx();
                    return;
                }
                initialize();
            }
    }

    @FXML
    public void Update(ActionEvent event) {
        int idB = 0;
        int userId = 1;
        int depId = -1;
        int userNumber = -1;
        int userWorkHours = -1;
        String userName = "";
        double userCapacity = -1.0;
        int c = 0;
        try {
            userId = Integer.parseInt(textId.getText());
            c++;
        } catch (NumberFormatException e) {
```

```
        }

        try {
            userName = textName.getText();
            if (userName.equals("")) throw new NumberFormatException();
            c++;
        } catch (NumberFormatException e) {
        }

        try {
            userCapacity = Double.parseDouble(textCapacity.getText());
            c++;
        } catch (NumberFormatException e) {
        }
        try {
            userNumber = Integer.parseInt(textNumber.getText());
            c++;
        } catch (NumberFormatException e) {
        }
        try {
            userWorkHours = Integer.parseInt(textWorkHours.getText());
            c++;
        } catch (NumberFormatException e) {
        }
        if (c == 0) {
            EventHandler.allEmptyFieldofUpdate();
            return;
        }
        int selectedIndex = tableUsers.getSelectionModel().getSelectedIndex();
        int usId = 0;
        try {
            usId = tableUsers.getItems().get(selectedIndex).getId();
        } catch (ArrayIndexOutOfBoundsException e) {
            EventHandler.rowNoSelected();
            return;
        }
        try {
            resultSet = null;
            resultSet = statement.executeQuery("select userId, userName, capacity,
departmentId, userNumber, workHours from user where userId=" + usId);
            while (resultSet.next()) {
                if (depId == -1) {
                    depId = resultSet.getInt(4);
                }

                if (userName.equals("")) {
                    userName += resultSet.getString(2);
                }

                if (userCapacity == -1.0) {
                    userCapacity = Double.parseDouble(resultSet.getString(3));
                }


                if (userNumber == -1) {
                    userNumber = resultSet.getInt(5);
                }
                if (userWorkHours == -1) {
                    userWorkHours = resultSet.getInt(6);
                }
            }
```

```
            statement.executeUpdate("update user set userId = " + userId + ", userName = '"
+ userName + "', capacity = " + userCapacity + ", departmentId = " + depId + ", userNumber =
" + userNumber + ", workHours = " + userWorkHours + " where userId = " + usId);
        } catch (SQLException e) {
            EventHandler.SQLEx();
            return;
        }
        initialize();
    }

    @FXML
    public void Delete(ActionEvent event) {
        int selectedIndex = tableUsers.getSelectionModel().getSelectedIndex();
        int userId = 0;
        try {
            userId = tableUsers.getItems().get(selectedIndex).getId();
        } catch (ArrayIndexOutOfBoundsException e) {
            EventHandler.rowNoSelected();
            return;
        }
        try {
            statement.executeUpdate("delete from user where userId = " + userId);
        } catch (SQLException e) {
            EventHandler.SQLEx();
            return;
        }
        initialize();
    }

    public void Clear(ActionEvent event) {
        try {
            statement.executeUpdate("delete from user");
        } catch (SQLException e) {
            EventHandler.SQLEx();
            return;
        }
        initialize();
    }
}
```

Додаток 3

Розробка програмного забезпечення для автоматизації енергоменеджменту на підприємстві на платформі ОС Windows

Опис програмного модулю

УКР.НТУУ“КПІ”.ТР61118_20Б 13-1

Аркушів 5

2020

АНОТАЦІЯ

Метою дипломної роботи є розробка програмного додатку, який дозволить користувачу вести облік електроенергії на підприємстві та вести політику енергоменеджменту.

Для розроблення додатку було використано мову програмування Java, базу даних створено засобами мови SQL та SQL Server. Інтерфейс налаштовано на платформі JavaFX за допомогою мов FXML та CSS.

Результатом роботи стало створення програмного продукту для моніторингу використання електроенергії на підприємстві.

ЗМІСТ

1. ВІДОМОСТІ ПРО ПРОГРАМНИЙ МОДУЛЬ

1.1 Опис логічної структури

Було розроблено кросплатформний додаток, який виконує функцію обліку використання електроенергії. Програма має вигляд віконного додатку і складається з трьох основних частин: бази даних, алгоритму обробки даних і користувацького інтерфейсу.

Програмний модуль потребує обов’язкового встановлення серверу бази даних, системи управління базами даних та спеціального драйверу підключення бази даних до програми.

Розроблений додаток отримує дані від користувача та створеної бази даних і формує звіти розрахунків витрат електроенергії у відділах та на підприємстві.

1.2 Вхідні та вихідні дані

Вхідними даними для системи є інформація про відділи підприємства, кількість, потужність і години роботи споживачів електроенергії у відділах.

Вихідними даними є звіти про витрати електроенергії у відділах та на підприємстві.

2. ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Програмний модуль було протестовано на персональному комп’ютері з операційною системою Windows, який працює на базі процесору 2.50 GHz Intel Core i5 та має 8 Гб оперативної пам’яті. Розроблене програмне забезпечення є кросплатформний, що дозволяє запускати його на комп’ютерах будь-якої потужності.